aws

开发人员指南

AWS Elastic Beanstalk

AWS Elastic Beanstalk: 开发人员指南

Table of Contents

什么是 AWS Elastic Beanstalk？

Amazon Web Services (AWS) 包含一百多种服务，每项服务都针对一个功能领域。服务的多样性可让您灵活地管理AWS基础设施，然而，判断应使用哪些服务以及如何进行预配置可能会非常困难。

借助 Elastic Beanstalk，您可以在AWS云中快速部署和管理应用程序，而不必了解运行这些应用程序的基础设施。Elastic Beanstalk 可降低管理的复杂性，但不会影响选择或控制。您只需上传应用程序，Elastic Beanstalk 将自动处理有关容量预配置、负载均衡、扩展和应用程序运行状况监控的部署细节。

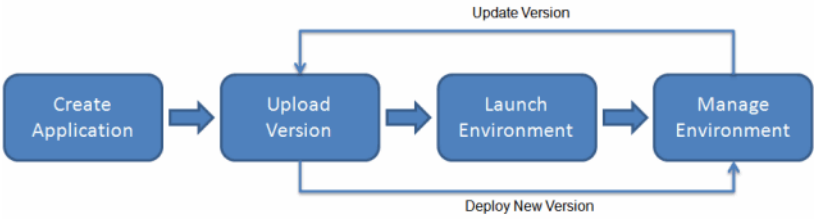
Elastic Beanstalk 支持在 Go、Java、.NET、Node.js、PHP、Python 和 Ruby 中开发的应用程序。在部署应用程序时，Elastic Beanstalk 会构建选定的受支持的平台版本，并预配置一个或多个AWS资源（如 Amazon EC2 实例）来运行应用程序。

您可通过使用 Elastic Beanstalk 控制台、AWS Command Line Interface (AWS CLI) 或 eb（专为 Elastic Beanstalk 设计的高级 CLI）与 Elastic Beanstalk 交互。

要了解有关如何使用 Elastic Beanstalk 部署示例 Web 应用程序的更多信息，请参阅 AWS 入门：部署 Web 应用程序。

您还可以直接从 Elastic Beanstalk Web 界面（控制台）执行大多数部署任务，如更改 Amazon EC2 实例队列的大小或监控应用程序。

要使用 Elastic Beanstalk，您需创建一个应用程序，将应用程序版本以应用程序源包的形式（如 Java .war 文件）上传到 Elastic Beanstalk，然后提供一些有关该应用程序的信息。Elastic Beanstalk 会自动启动环境，然后创建并配置运行代码所需的AWS资源。启动环境后，您即可管理环境并部署新应用程序版本。下图说明了 Elastic Beanstalk 的工作流程。



创建并部署应用程序后，可通过 Elastic Beanstalk 控制台、API 或命令行界面（包括统一的 AWS CLI）查看有关应用程序的信息，包括指标、事件和环境状态。

定价

Elastic Beanstalk 不收取额外费用。您只需为应用程序使用的基础AWS资源付费。有关定价的详细信息，请参阅 Elastic Beanstalk 服务详细信息页面。

后续工作

本指南包含有关 Elastic Beanstalk Web 服务的概念性信息，以及有关如何使用该服务部署 Web 应用程序的信息。各个部分介绍了如何使用 Elastic Beanstalk 控制台、命令行界面 (CLI) 工具和 API 部署和管理 Elastic Beanstalk 环境。本指南还介绍了 Elastic Beanstalk 如何与 Amazon Web Services 提供的其他服务集成。

我们建议您先阅读开始使用 Elastic Beanstalk，了解如何开始使用 Elastic Beanstalk。入门 会一步一步地引导您创建、查看和更新 Elastic Beanstalk 应用程序，以及编辑和终止 Elastic Beanstalk 环境。入门 还介绍了访问 Elastic Beanstalk 的不同方法。

要了解有关 Elastic Beanstalk 应用程序及其组件的更多信息，请参阅以下页面。

- Elastic Beanstalk 概念
- Elastic Beanstalk 平台词汇表
- Elastic Beanstalk 平台维护的责任共担模型
- Elastic Beanstalk 平台支持策略

开始使用 Elastic Beanstalk

为了帮助您了解 AWS Elastic Beanstalk 工作原理，本教程将引导您创建、探索、更新和删除 Elastic Beanstalk 应用程序。此教程可在 1 小时内完成。

使用 Elastic Beanstalk 不收取任何费用， AWS 但它为本教程创建的资源是实时的（并且不在沙盒中运行）。您将为这些资源支付标准使用费，直到您在本教程最后终止这些资源为止。总费用一般少于一美元。有关如何最大限度减少费用的信息，请参阅 AWS 免费套餐。

主题

设置：创建 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

步骤 1：创建示例应用程序

在此步骤中，您将从先前存在的示例应用程序开始创建新应用程序。Elastic Beanstalk 支持针对不同的编程语言、应用程序服务器以及 Docker 容器的平台。您在创建应用程序时选择一个平台。

创建应用程序和环境

要创建您的示例应用程序，您将使用 Create application（创建应用程序）控制台向导。它创建 Elastic Beanstalk 应用程序并在其中启动一个环境。环境是运行应用程序代码所需的 AWS 资源集合。

创建示例应用程序

1. 打开 Elastic Beanstalk 控制台。
2. 选择创建应用程序。
3. 对于 Application name（应用程序名称），输入 **`getting-started-app`**。
4. （可选）添加应用程序标签。
5. 对于 Platform（平台），选择一个平台。
6. 选择下一步。
7. 这时将显示配置服务访问权限页面。
8. 对于服务角色，请选择使用现有的服务角色。
9. 下面我们将重点介绍 EC2 实例配置文件下拉列表。此下拉列表中显示的值可能因您的账户之前是否创建过新环境而异。

   根据列表中显示的值，选择以下选项中的一个。

   - 如果在下拉列表中显示有 `aws-elasticbeanstalk-ec2-role`，请从 EC2 实例配置文件下拉列表中将其选中。
   - 如果列表中显示的是其他值，并且这是您环境的默认 EC2 实例配置文件，请从 EC2 实例配置文件下拉列表中选中该值。
   - 如果 EC2 实例配置文件下拉列表未显示任何可供选择的值，请按下面为 EC2 实例配置文件创建 IAM 角色的过程操作。

完成为 EC2 实例配置文件创建 IAM 角色中的步骤，以创建一个之后为 EC2 实例配置文件选择的 IAM 角色。然后返回此步骤。

现在您已创建了一个 IAM 角色并刷新了列表，该角色将在下拉列表中显示为一个选项。从 EC2 实例配置文件下拉列表中选中您刚刚创建的 IAM 角色。

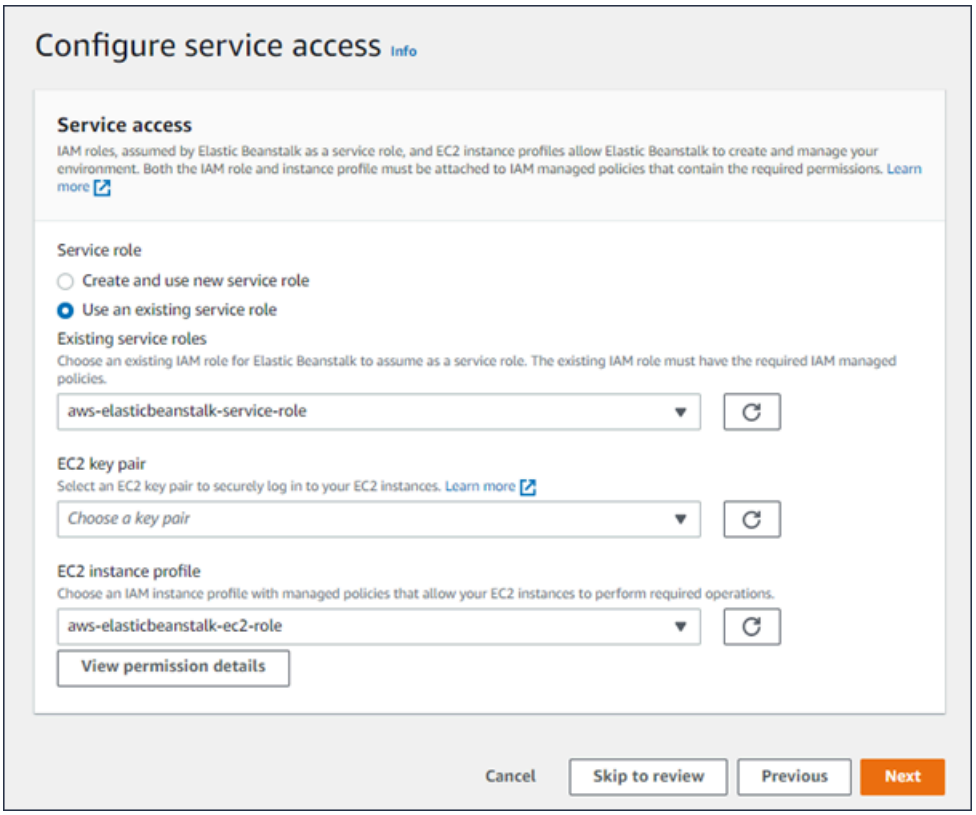
10. 在 Configure service access（配置服务访问）页面上选择 Skip to Review（跳至审核）。

    这将跳过可选步骤。

11. Review（审核）页面将显示所有选择的摘要。

    在页面底部选择 Submit（提交）。

为 EC2 实例配置文件创建 IAM 角色



创建用于 EC2 实例配置文件选择的 IAM 角色

1. 选择查看权限详细信息。这将在 EC2 实例配置文件下拉列表下显示。

这时会显示一个名为查看实例配置文件权限的模态窗口。此窗口将列出您需要附加到所创建的新 EC2 实例配置文件的托管式配置文件。此外还提供了一个用于启动 IAM 控制台的链接。

2. 选择窗口顶部显示的 IAM 控制台链接。
3. 请在 IAM 控制台的导航窗格中，选择 Roles（角色）。
4. 选择 创建角色。
5. 在可信实体类型下，选择 AWS 服务。
6. 在 Use case（使用案例）下，选择 EC2。
7. 选择下一步。
8. 附加适当的托管式策略。滚动查看实例配置文件权限模式窗口，以查看托管式策略。这些策略还将在此处列出：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 选择下一步。
10. 输入角色的名称。
11. （可选）将标签添加到角色。
12. 选择 创建角色。
13. 返回已打开的 Elastic Beanstalk 控制台窗口。
14. 关闭查看实例配置文件权限模态窗口。

> **Important**
>
> 不要关闭显示 Elastic Beanstalk 控制台的浏览器页面。

15. 选择 EC2 实例配置文件下拉列表旁边的  （刷新）。

    这将刷新下拉列表，以确保您刚刚创建的角色会在下拉列表中显示。

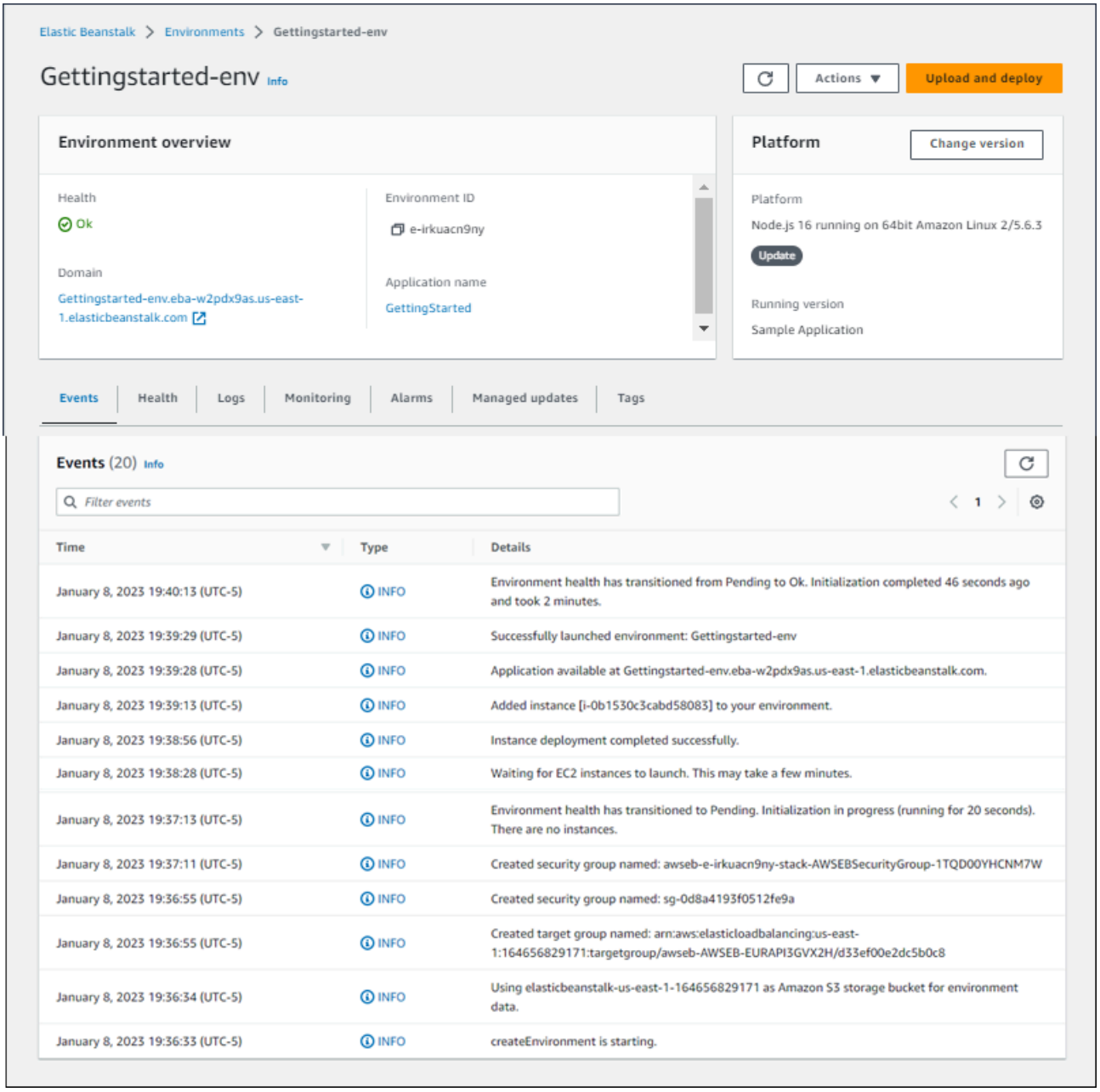
Elastic Beanstalk 工作流

为了在 AWS 资源上部署和运行示例应用程序，Elastic Beanstalk 会执行以下操作。这些操作需要约 5 分钟的时间完成。

1. 创建名为的 Elastic Beanstalk 应用程序getting-started-app。
2. 使用以下 AWS 资源启动名为 GettingStartedApp-env 的环境：
   - Amazon Elastic Compute Cloud (Amazon EC2) 实例（虚拟机）
   - Amazon EC2 安全组
   - 一个 Amazon Simple Storage Service (Amazon S3) 存储桶
   - 亚马逊 CloudWatch 警报
   - 一个 AWS CloudFormation 堆栈
   - 域名

   有关这些 AWS 资源的详细信息，请参阅the section called “AWS 为示例应用程序创建的资源”。
3. 创建一个名为 Sample Application 的新应用程序版本。这是默认的 Elastic Beanstalk 示例应用程序文件。
4. 将示例应用程序的代码部署到 GettingStartedApp-en v 环境中。

在创建环境过程中，控制台将跟踪进度并显示事件。



当所有资源启动并且运行应用程序的 EC2 实例通过运行状况检查后，环境的运行状况将变为 0k。现在，您可以使用您的 Web 应用程序的网站。

AWS 为示例应用程序创建的资源

在创建示例应用程序时，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。

- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

步骤 2：探索您的环境

要查看 Elastic Beanstalk 应用程序的环境概述，请使用 Elastic Beanstalk 控制台中的 Environment overview（环境概述）页面。

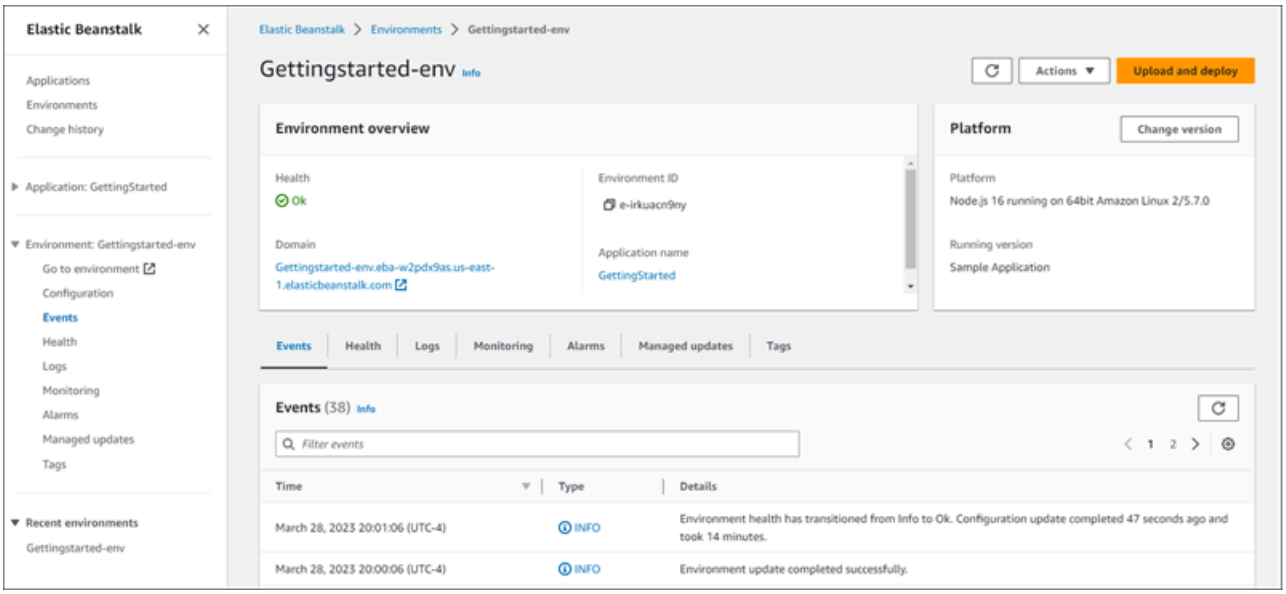
查看环境概述

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

Environment overview（环境概述）页面的上半部分显示有关环境的顶级信息。这包括其名称、域 URL、当前运行状况、当前部署的应用程序版本的名称以及运行应用程序的平台版本。在概述窗格下方，您可以在 Events（事件）选项卡中查看最新的环境事件。其他选项卡显示有关环境的其他主要详细信息。

要了解有关环境层、平台、应用程序版本和其他 Elastic Beanstalk 概念的更多信息，请参阅概念。



当 Elastic Beanstalk AWS 创建您的资源并启动您的应用程序时，环境处于状态。`Pending`有关启动事件的状态消息会不断地添加到概述中。

环境的域或 URL 位于 Environment overview（环境概述）页面的上半部分，环境的 Health（运行状况）下方。这是环境正在运行的 Web 应用程序的 URL。选择此 URL 以转到示例应用程序的祝贺页面。左侧的导航窗格列出了 Go to environment（转到环境）链接，用于启动相同的应用程序页面。

左侧导航窗格中还列出了 Configuration（配置），它显示了 Configuration overview（配置概述）页面。此页面显示环境配置选项值摘要，按类别分组。

该页面下半部分显示的选项卡包含有关环境的更多详细信息，并提供对其他功能的访问：

- Events (事件) - 显示来自 Elastic Beanstalk 服务以及此环境使用其资源的其他服务的信息或错误消息。

- Health (运行状况) - 显示有关运行应用程序的 Amazon EC2 实例的状态和详细运行状况信息。
- Logs（日志）– 从您的环境中的 Amazon EC2 检索和下载日志。您可以检索完整的日志或最近的活动。检索到的日志在 15 分钟内可用。
- Monitoring (监控) - 显示环境的统计数据（如平均延迟和 CPU 使用率）。
- Alarms（警报）– 显示您为环境指标配置的警报。您可以在此页面上添加、修改或删除警报。
- Managed updates（托管更新）– 显示有关即将到来和已完成的托管平台更新以及实例替换的信息。
- Tags (标签) - 显示环境标签，并允许您对其进行管理。标签是应用于您的环境的密钥值对。

> **Note**
>
> 控制台左侧的导航窗格列出了与选项卡同名的链接。选择这些链接中的任何一个都将显示相应选项卡的内容。

步骤 3：部署应用程序的新版本

您可能需要定期部署新版本的应用程序。您可以随时部署新版本，前提是在环境中没有进行其他更新操作。

您开始使用本教程的应用程序版本称为 Sample Application (示例应用程序)。

更新您的应用程序版本

1. 下载与环境的平台相匹配的示例应用程序。使用以下任一应用程序。

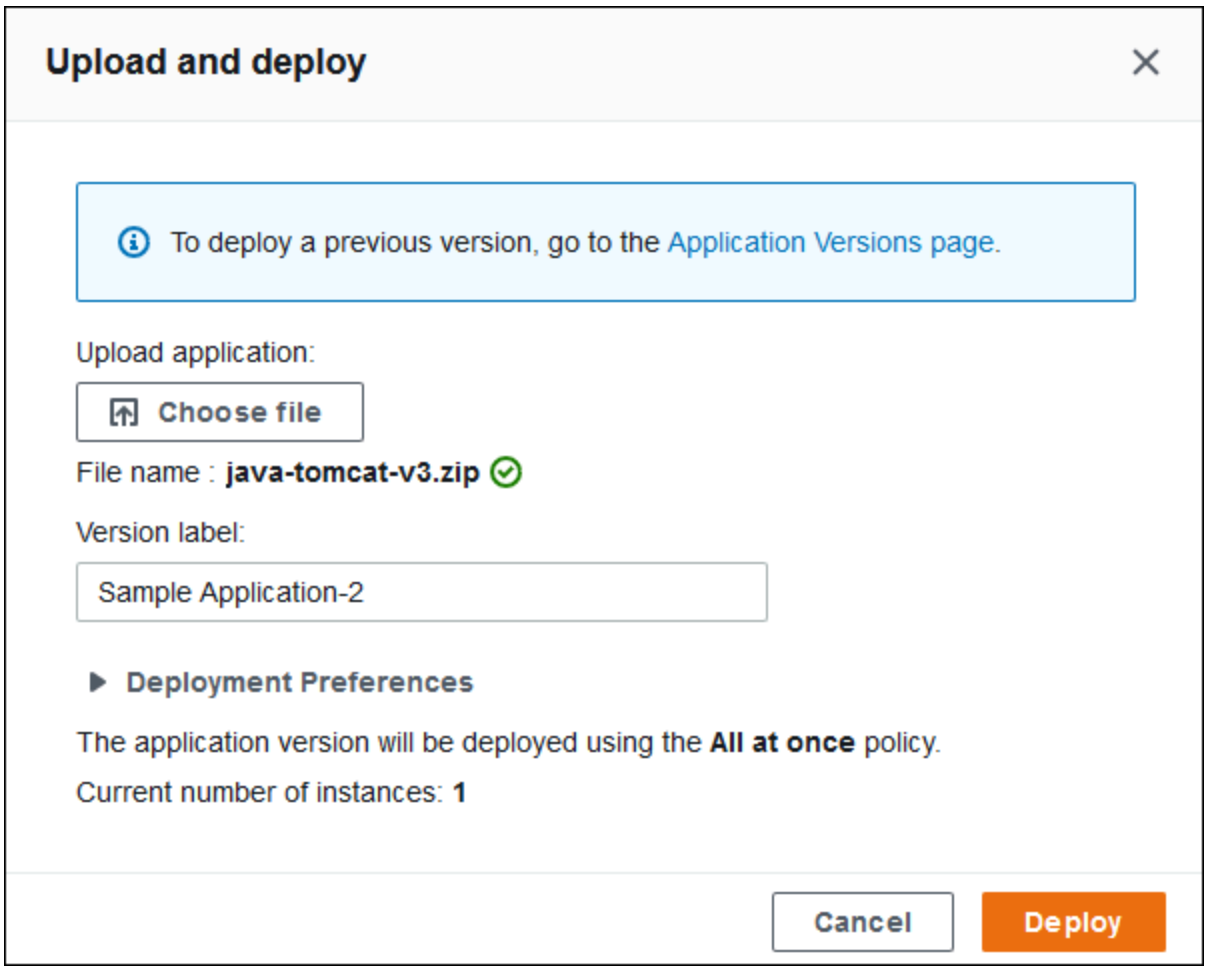
   - Docker – docker.zip
   - 多容器 Docker — 2.zip docker-multicontainer-v
   - 预配置的 Docker (Glassfish) — 1.zip docker-glassfish-v
   - Go – go.zip
   - Corretto – corretto.zip
   - Tomcat – tomcat.zip
   - Linux 上的.NET 内核 — dotnet-core-linux.zi
   - .NET 核心 — dotnet-asp-windows.zip
   - Node.js – nodejs.zip
   - PHP – php.zip

- Python – python.zip
- Ruby – ruby.zip

2. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
3. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

4. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
5. 选择 Choose file (选择文件)，然后上传您已下载的示例应用程序源包。



控制台会自动使用新的唯一标签填充 Version label (版本标签)。如果您键入自己的版本标签，请确保它是唯一的。

6. 选择部署。

当 Elastic Beanstalk 将文件部署到 Amazon EC2 实例时，您可以在环境概述中查看部署状态。应用程序版本更新时，Environment Health (环境运行状况) 状态为灰色。部署完成时，Elastic Beanstalk 会执行应用程序运行状况检查。当应用程序对运行状况检查进行响应时，它被视为运行状况良好，状态会变回绿色。环境概述显示新的 Running Version (运行版本) - 您作为 Version label (版本标签) 提供的名称。

Elastic Beanstalk 也会上传您的新应用程序版本并将其添加到应用程序版本表。要查看该表，请选择导航窗格getting-started-app下方的应用程序版本。

步骤 4：配置环境

您可以配置环境，使其更适合您的应用程序。例如，如果您的应用程序需要进行大量计算，那么您可以更改正在运行您的应用程序的 Amazon Elastic Compute Cloud (Amazon EC2) 实例类型。为了应用配置更改，Elastic Beanstalk 将执行环境更新。

有一些配置更改很简单，而且会即刻生效。有些更改需要删除和重新创建 AWS 资源，这可能需要几分钟。更改配置设置时，Elastic Beanstalk 会向您发出有关潜在应用程序停机的警告。

进行配置更改

在此配置更改示例中，您将编辑环境的容量设置。您配置负载均衡的可扩展环境（该环境在其 Auto Scaling 组中有 2 到 4 个 Amazon EC2 实例），然后验证是否已发生更改。Elastic Beanstalk 会创建一个额外的 Amazon EC2 实例，并将其添加到它最初创建的单个实例中。然后，Elastic Beanstalk 将这两个实例与环境的负载均衡器关联。这样，可以提高应用程序的响应速度，并提高其可用性。

更改环境的容量

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Instance traffic and scaling（实例流量和扩展）配置类别中，选择 Edit（编辑）。
5. 折叠 Instances（实例）部分，这样您就可以更轻松地看到 Capacity（容量）部分。在 Auto Scaling group（自动扩缩组）下方，将 Environment type（环境类型）更改为 Load balanced（负载均衡）。

6. 在实例行上，将最大更改为 **4**，然后将最小更改为 **2**。
7. 要保存更改，请选择页面底部的 Apply（应用）。
8. 将显示警告信息，告诉您此更新将替换您当前的所有实例。选择确认。
9. 将显示 Environment overview（环境概述）页面，显示 Events（事件）选项卡。

   环境更新可能需要几分钟的时间。要确定它已完成，请在事件列表中查找事件 Successfully deployed new configuration to environment (已成功将新配置部署到环境)。这确认了 Auto Scaling 实例的最少数目已经设置为 2。Elastic Beanstalk 将自动启动第二个实例。

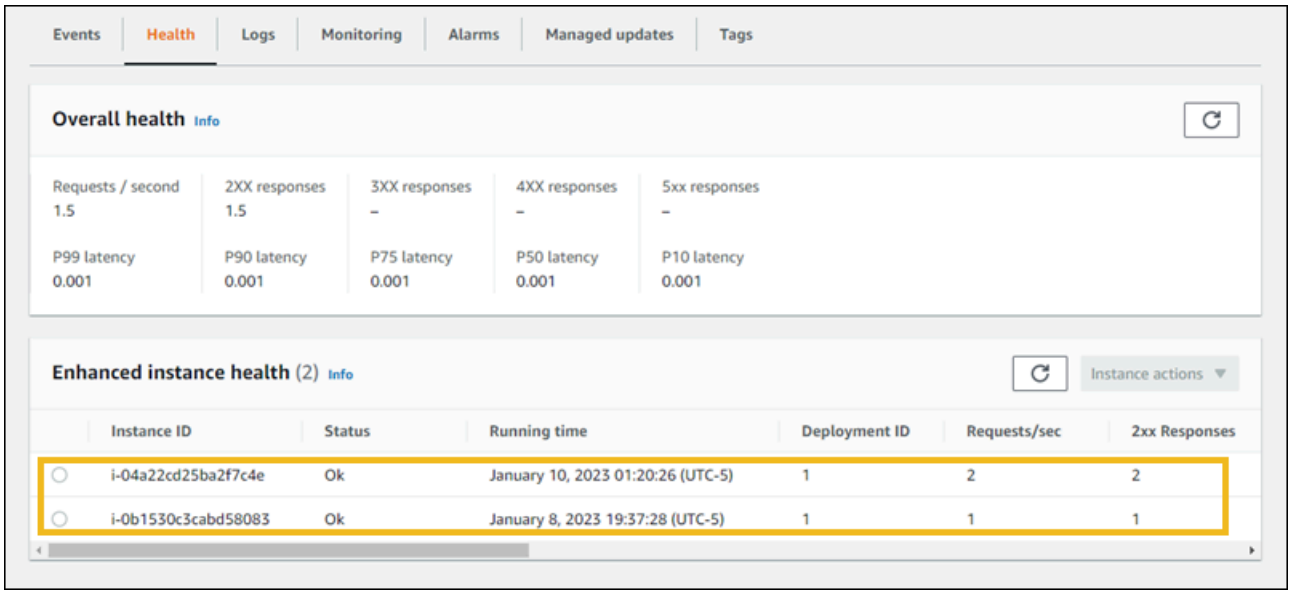
验证配置更改

环境更新完成且环境已准备就绪时，请验证您的更改。

验证增加的容量

1. 从选项卡列表或左侧导航窗格中选择 Health（运行状况）。
2. 查看 Enhanced instance health（增强型实例运行状况）部分。

   您可以看到列出了两个 Amazon EC2 实例。您的环境容量已提高到两个实例。



第 5 步：清理

恭喜您！您已成功将示例应用程序部署到 AWS 云端，上传了新版本，并修改了其配置以添加第二个 Auto Scaling 实例。为确保您不用为未使用的任何服务付费，请删除所有应用程序版本并终止环境。这还会删除环境为您创建的 AWS 资源。

删除应用程序及所有关联资源

1. 删除所有应用程序版本。
   a. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
   b. 在导航窗格中，选择 “应用程序”，然后选择getting-started-app。
   c. 在导航窗格中，找到应用程序的名称，然后选择 Application versions（应用程序版本）。
   d. 在 Application versions (应用程序版本) 页面上，选择要删除的所有应用程序版本。
   e. 选择操作，然后选择删除。
   f. 启用 Delete versions from Amazon S3 (从 Amazon S3 删除版本)。
   g. 选择 Delete (删除)，然后选择 Done (完成)。
2. 终止环境。
   a. 在导航窗格中，选择 getting-started-app，然后在环境GettingStartedApp列表中选择-env。
   b. 选择 Actions（操作），然后选择 Terminate Environment（终止环境）。
   c. 键入环境名称确认要终止 GettingStartedApp-env，然后选择终止。
3. 删除 getting-started-app 应用程序。
   a. 在导航窗格中，选择getting-started-app。
   b. 选择 Actions （操作)，然后选择 Delete application （删除应用程序)。
   c. 键入应用程序名称确认要删除 getting-started-app，然后选择删除。

后续步骤

现在您已知道如何创建 Elastic Beanstalk 应用程序和环境，我们建议您阅读概念。本主题提供有关 Elastic Beanstalk 组件和架构的信息，并介绍 Elastic Beanstalk 应用程序的重要设计注意事项。

除了 Elastic Beanstalk 控制台以外，您还可以使用以下工具创建和管理 Elastic Beanstalk 环境。

EB CLI

EB CLI 是用于创建和管理环境的命令行工具。有关详细信息，请参阅 使用 Elastic Beanstalk 命令行界面 (EB CLI)。

AWS SDK for Java

AWS SDK for Java 提供了一个 Java API，您可以使用它来构建使用 AWS 基础设施服务的应用程序。借助 AWS SDK for Java，您可以在几分钟内开始使用一个可下载的软件包，其中包含 AWS Java 库、代码示例和文档。

AWS SDK for Java 需要 J2SE 开发套件 5.0 或更高版本。您可以从 http://developers.sun.com/downloads/ 下载最新的 Java 软件。该开发工具包还要求有 Apache Commons (Codec、HTTPClient 和 Logging)，以及 Saxon-HE 第三方包，这个第三方包位于该开发工具包的第三方目录下。

有关更多信息，请参阅适用于 Java 的AWS SDK。

AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 是 Eclipse Java IDE 的开源插件。您可以使用它来创建使用预配置的 AWS Java Web 项目 AWS SDK for Java，然后将 Web 应用程序部署到 Elastic Beanstalk。Elastic Beanstalk 插件构建在 Eclipse Web Tools Platform (WTP) 之上。该工具包提供 Travel Log 示例 Web 应用程序模板，该模板演示如何使用 Amazon S3 和 Amazon SNS。

为确保您拥有全部 WTP 依赖项，建议您从 Eclipse 的 Java EE 分发版开始。您可以从 http://eclipse.org/downloads/ 下载它。

有关使用适用于 Eclipse 的 Elastic Beanstalk 插件的更多信息，请参阅 AWS Toolkit for Eclipse。要开始使用 Eclipse 创建 Elastic Beanstalk 应用程序，请参阅在 Elastic Beanstalk 上创建和部署 Java 应用程序。

AWS SDK for .NET

AWS SDK for .NET 使您能够构建使用 AWS 基础设施服务的应用程序。借助 AWS SDK for .NET，您可以在几分钟内开始使用单个可下载的软件包，其中包括 AWS .NET 库、代码示例和文档。

有关更多信息，请参阅 AWS SDK for .NET。有关支持的 .NET Framework 和 Visual Studio 版本，请参阅 AWS SDK for .NET 开发人员指南。

AWS Toolkit for Visual Studio

使用该 AWS Toolkit for Visual Studio 插件，您可以将现有的.NET 应用程序部署到 Elastic Beanstalk。您也可以使用预先配置的 AWS 模板来创建项目。 AWS SDK for .NET

有关前提条件和安装信息，请参阅 AWS Toolkit for Visual Studio。要开始使用 Visual Studio 创建 Elastic Beanstalk 应用程序，请参阅在 Elastic Beanstalk 上创建和部署.NET Windows 应用程序。

AWS Node.js JavaScript 中的 SDK

Node.js JavaScript 中的 S AWS DK 使您能够在 AWS 基础架构服务之上构建应用程序。使用 Node.js JavaScript 中的 S AWS DK，您可以在几分钟内开始使用一个可下载的软件包，其中包含 AWS Node.js 库、代码示例和文档。

有关更多信息，请参阅 Node.js JavaScript 中的AWS 软件开发工具包。

AWS SDK for PHP

AWS SDK for PHP 使您能够在 AWS 基础架构服务之上构建应用程序。有了这个 AWS SDK for PHP，你可以在几分钟内开始使用一个可下载的软件包，其中包含 AWS PHP 库、代码示例和文档。

AWS SDK for PHP 需要 PHP 5.2 或更高版本。有关下载详细信息，请参阅 http://php.net/。

有关更多信息，请参阅适用于 PHP 的AWS SDK。

AWS SDK for Python (Boto)

有了这个 AWS SDK for Python (Boto)，你可以在几分钟内开始使用一个可下载的软件包，其中包含 AWS Python 库、代码示例和文档。您可以在 API 之上构建 Python 应用程序，从而免去了直接对 Web 服务接口编写代码的麻烦。

该 all-in-one 库提供了 Python 开发人员友好型 API，这些API隐藏了许多与 AWS 云端编程相关的较低级别任务，包括身份验证、请求重试和错误处理。对于如何使用该库构建应用程序，该开发工具包提供了用 Python 编写的实用示例。

有关 Boto、示例代码、文档、工具和其他资源的信息，请参阅 Python 开发人员中心。

AWS SDK for Ruby

您可以在几分钟内开始使用一个可下载的软件包，其中包含 AWS Ruby 库、代码示例和文档。您可以在 API 的顶层构建 Ruby 应用程序，从而免去了直接对照 Web 服务界面编码的麻烦。

该 all-in-one 库提供了 Ruby 开发人员友好型 API，这些API隐藏了许多与 AWS 云端编程相关的较低级别任务，包括身份验证、请求重试和错误处理。对于如何使用该库构建应用程序，该开发工具包提供了用 Ruby 编写的实用示例。

有关开发工具包、示例代码、文档、工具和其他资源的信息，请参阅 Ruby 开发人员中心。

Elastic Beanstalk 概念

AWS Elastic Beanstalk 允许您管理将应用程序作为环境运行的所有资源。以下是一些关键的 Elastic Beanstalk 概念。

应用程序

Elastic Beanstalk 应用程序 是 Elastic Beanstalk 组件的逻辑集合，包括环境、版本 和环境配置。在 Elastic Beanstalk 中，应用程序在概念上类似于文件夹。

应用程序版本

在 Elastic Beanstalk 中，应用程序版本 指的是 Web 应用程序的可部署代码的特定标记迭代。一个应用程序版本指向一个包含可部署代码（例如，Java WAR 文件）的 Amazon Simple Storage Service (Amazon S3) 对象。应用程序版本是应用程序的组成部分。应用程序可以有多个版本，每个应用程序版本都是唯一的。在运行环境中，您可以部署已上传到应用程序的任意应用程序版本，也可以上传并立即部署新的应用程序版本。您可以上传多个应用程序版本，以测试 Web 应用程序不同版本之间的差异。

环境

环境是运行应用程序版本的AWS资源的集合。每个环境一次只运行一个应用程序版本，但您可以同时在多个环境中运行相同或不同的应用程序版本。当您创建环境时，Elastic Beanstalk 会预配置运行您指定的应用程序版本所需的资源。

环境套餐

在启动 Elastic Beanstalk 环境时，您需首先选择环境层。环境层指定环境运行的应用程序类型，并确定 Elastic Beanstalk 预配置哪些资源来支持这种类型。为 HTTP 请求提供服务的应用程序在 Web 服务器环境层中运行。从 Amazon Simple Queue Service (Amazon SQS) 队列中提取任务的后端环境在工作线程环境层中运行。

环境配置

环境配置 标识一组参数和配置，这些参数和配置用于定义环境及其相关资源的行为方式。当您更新环境的配置设置时，Elastic Beanstalk 会自动将更改应用到现有资源，或者删除现有资源并部署新资源（取决于更改的类型）。

已保存的配置

保存的配置 是一种模板，您可以将其用作创建独特环境配置的起点。您可以使用 Elastic Beanstalk 控制台、EB CLI、AWS CLI 或 API 创建配置和修改已保存的配置，以及将配置应用于环境。API 和 AWS CLI 将保存的配置称为配置模板。

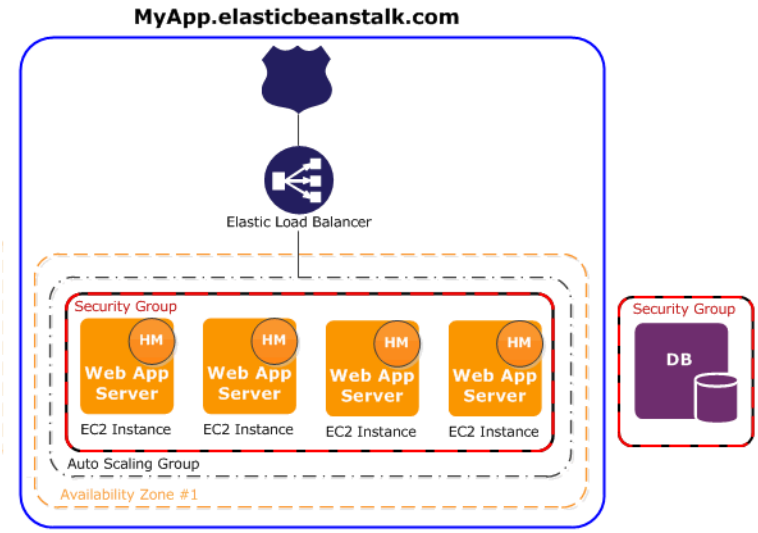
平台

平台 是操作系统、编程语言运行时、Web 服务器、应用程序服务器和 Elastic Beanstalk 组件的组合。您可以设计 Web 应用程序并将其定向到平台。Elastic Beanstalk 提供各种可供您构建应用程序的平台。

有关详细信息，请参阅 Elastic Beanstalk 平台。

Web 服务器环境

下图显示了一个 Web 服务器环境层的示例 Elastic Beanstalk 架构，并显示了这种环境层中各个组件协同工作的方式。



环境是应用程序的核心。在图中，环境显示在顶层实线内。当您创建环境时，Elastic Beanstalk 会预配置运行应用程序所需的资源。为环境创建的AWS资源包括一个 Elastic Load Balancer（图中的 ELB）、一个 Auto Scaling 组和一或更多个 Amazon Elastic Compute Cloud (Amazon EC2) 实例。

每个环境有一个指向负载均衡器的别名记录(URL)。该环境有一个 URL，例如 `myapp.us-west-2.elasticbeanstalk.com`。通过使用别名记录，此 URL 在 Amazon

Route 53 中的别名为 Elastic Load Balancing URL（例如 `abcdef-123456.us-west-2.elb.amazonaws.com`）。Amazon Route 53 是一种可用性高、可扩展性强的域名系统 (DNS) Web 服务。它可以向您的基础设施提供安全可靠的路由。您通过 DNS 提供商注册的域名会将请求转发到别名记录。

负载均衡器位于 Amazon EC2 实例的前面，后者是 Auto Scaling 组的一部分。Amazon EC2 Auto Scaling 可自动启动其他 Amazon EC2 实例，以适应应用程序上增大的负载。如果应用程序上的负载减小，Amazon EC2 Auto Scaling 会终止实例，但始终会至少保留一个正在运行的实例。

在 Amazon EC2 实例上运行的软件栈取决于容器类型。容器类型定义的是将在该环境中使用的基础设施拓扑和软件栈。例如，包含 Apache Tomcat 容器的 Elastic Beanstalk 环境使用 Amazon Linux 操作系统、Apache Web 服务器和 Apache Tomcat 软件。有关所支持的容器类型的列表，请参阅 Elastic Beanstalk 支持的平台。每个运行应用程序的 Amazon EC2 实例都使用这些容器类型之一。此外，名为主机管理器 (HM) 的软件组件会在每个 Amazon EC2 实例上运行。主机管理器负责以下内容：

- 部署应用程序
- 汇总事件和指标，以通过控制台、API 或命令行进行检索
- 生成实例级事件
- 监控应用程序日志文件中是否有关键错误
- 监控应用程序服务器
- 修补实例组件
- 交替您的应用程序日志文件，并将它们发布到 Amazon S3

主机管理器报告指标、错误、事件和服务器实例状态，它们是通过 Elastic Beanstalk 控制台、API 和 CLI 获取的。

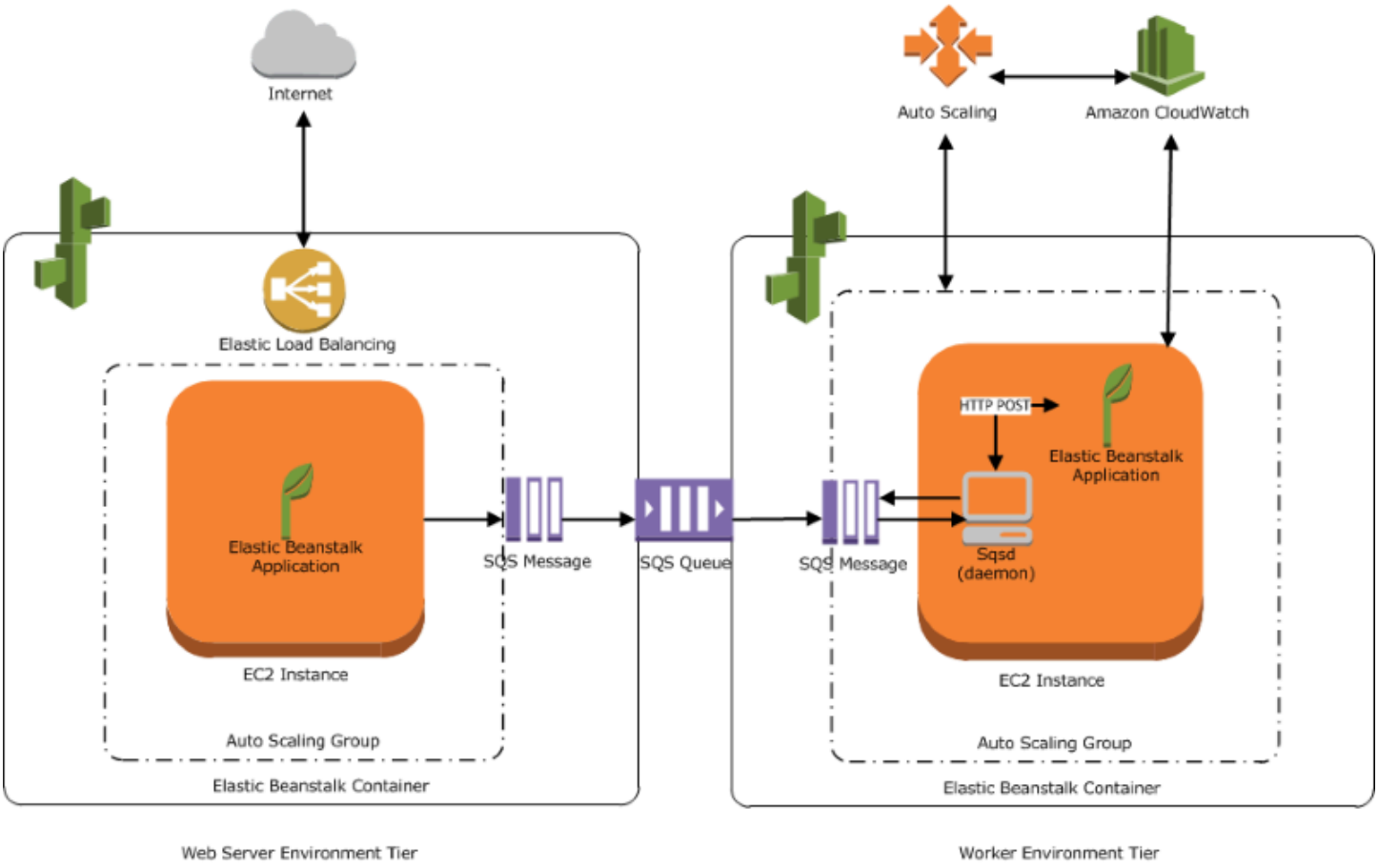
图中显示的 Amazon EC2 实例是一个安全组 的一部分。安全组定义实例的防火墙规则。默认情况下，Elastic Beanstalk 定义一个安全组，该安全组允许每个人都可以使用端口 80 (HTTP) 进行连接。您可以定义一个以上的安全组。例如，您可以为您的数据库服务器定义一个安全组。有关 Amazon EC2 安全组及如何为您的 Elastic Beanstalk 应用程序配置这些安全组的更多信息，请参阅安全组。

工作线程环境

为工作线程环境层创建的AWS资源包括一个 Auto Scaling 组、一个或多个 Amazon EC2 实例和一个 IAM 角色。对于工作线程环境层，Elastic Beanstalk 还会创建并预配置一个 Amazon SQS 队列（如果您还没有这样一个队列）。启动工作线程环境时，Elastic Beanstalk 会根据您选择的编程语言安装必要

的支持文件并在 Auto Scaling 组中的每个 EC2 实例上安装一个守护程序。守护程序从 Amazon SQS 队列中读取消息。守护程序将其读取的每条消息中的数据发送到在工作线程环境中运行的 Web 应用程序以进行处理。如果您的工作线程环境中有多个实例，则每个实例都有自己的守护程序，但它们都从同一 Amazon SQS 队列中读取数据。

下图显示了不同的组件及其跨环境和AWS服务的交互情况。



Amazon CloudWatch 用于警报和运行状况监控。有关更多信息，请转至 基本运行状况报告。

有关工作线程环境层的工作方式的详细信息，请参阅Elastic Beanstalk 工作线程环境。

设计注意事项

因为使用 AWS Elastic Beanstalk 部署的应用程序在 AWS Cloud 资源上运行，所以您在设计应用程序时应注意几个方面：可扩展性、安全性、持久性存储、容错能力、内容分发、软件更新和补丁和连接性。本主题分别介绍了其中的每一项。有关涵盖架构以及安全性和经济因素等主题的技术 AWS 白皮书的完整列表，请转到 AWS 云计算白皮书。

可扩展性

与云环境相对，在物理硬件环境中运行时，您可以通过两种方式之一实现可扩展性。您可以通过垂直缩放纵向扩展，也可以通过水平缩放进行横向扩展。纵向扩展方法要求您投资强大的硬件，这些硬件可以支持业务不断增长的需求。横向扩展方法要求您遵循分布式投资模式。因此，您的硬件和应用程序获取可以更有针对性，数据集是联合的，而且您的设计以服务为导向。纵向扩展方法可能会非常昂贵，并且仍然存在需求可能会超出容量的风险。在这方面，横向扩展方法通常更有效。但是，在使用它时，您必须能够定期预测需求，并分块部署基础设施以满足该需求。因此，此方法往往导致未用容量，可能需要仔细监控。

通过迁移到云，您可以充分利用云的弹性，使您的基础设施与需求很好地结合起来。弹性有助于简化资源获取和发布。有了它，您的基础设施可随着需求的波动而迅速横向缩减和横向扩展。要使用它，请配置您的 Auto Scaling 设置，使之根据环境中的资源指标扩展和收缩。例如，您可以设置诸如服务器利用率或网络输入/输出等指标。您可以使用 Auto Scaling 来实现计算容量，以便在使用量上升时自动添加，并在使用量下降时将其删除。您可以将系统指标（例如 CPU、内存、磁盘输入/输出和网络输入/输出）发布到 Amazon CloudWatch。然后，您可以使用 CloudWatch 配置警报以触发 Auto Scaling 操作或根据这些指标发送通知。有关如何配置 Auto Scaling 的说明，请参阅 Elastic Beanstalk 环境的 Auto Scaling 组。

我们还建议您将所有 Elastic Beanstalk 应用程序设计地尽可能保持无状态，以便使用可根据需要横向扩展的、松散耦合的容错组件。有关为 AWS 设计可扩展应用程序架构的更多信息，请参阅 AWS Well-Architected Framework。

安全性

在 AWS 上，安全性为责任共担。Amazon Web Services 负责保护您的环境中的物理资源，并确保云是可供您运行应用程序的安全场所。您负责保证进出您的 Elastic Beanstalk 环境的数据的安全，以及您的应用程序的安全。

要保护在您的应用程序和客户端之间流动的信息，请配置 SSL。要配置 SSL，您需要来自 AWS Certificate Manager (ACM) 的免费证书。如果您已经拥有一个来自外部证书颁发机构 (CA) 的证书，则可以使用 ACM 导入您的证书。或者，您还可以使用 AWS CLI 导入该证书。

如果 ACM 不在您的AWS 区域中可用，则可以从外部 CA（例如 VeriSign 或 Enust）购买证书。然后，使用 AWS Command Line Interface (AWS CLI) 将第三方或自签名证书和私有密钥上载到 AWS Identity and Access Management (IAM)。证书的公有密钥将您的服务器对浏览器进行身份验证。它还用做创建加密双向数据的共享会话密钥的基础。有关如何创建、上传和分配 SSL 证书到您的环境的说明，请参阅 为 Elastic Beanstalk 环境配置 HTTPS。

为您的环境配置 SSL 证书后，数据在客户端和您环境的 Elastic Load Balancing 负载均衡器之间加密传输。默认情况下，加密在负载均衡器处终止，负载均衡器与 Amazon EC2 实例之间的通信是未加密的。

持久性存储

Elastic Beanstalk 应用程序在没有持久性本地存储的 Amazon EC2 实例上运行。Amazon EC2 实例终止时，不会保存本地文件系统。新的 Amazon EC2 实例从默认文件系统开始。我们建议您将应用程序配置为在持久性数据源中存储数据。AWS 提供了大量持久性存储服务，您可将它们用于应用程序。下表列出了这些版本。

存储服务	服务文档	Elastic Beanstalk 集成
Amazon S3	Amazon Simple Storage Service 文档	将 Elastic Beanstalk 和 Amazon S3 结合使用
Amazon Elastic File System	Amazon Elastic File System 文档	配合使用 Elastic Beanstalk 和 Amazon Elastic File System
Amazon Elastic Block Store	Amazon Elastic Block Store 功能指南：Elastic Block Store	
Amazon DynamoDB	Amazon DynamoDB 文档	配合使用 Elastic Beanstalk 和 Amazon DynamoDB
Amazon Relational Database Service (RDS)	Amazon Relational Database Service 文档	将 Elastic Beanstalk 和 Amazon RDS 结合使用

> **Note**
>
> Elastic Beanstalk 为您创建了 webapp 用户，您可以以应用程序目录所有者的身份在 Amazon EC2 实例上设置它。对于 2022 年 2 月 3 日或之后发布的 Amazon Linux 2 平台版本，Elastic Beanstalk 会针对新环境为 webapp 用户分配一个 uid（用户 ID）和 900 的 gid（组 ID）值。对于平台版本更新后的现有环境，它也是如此。这种方法保持了 webapp 用户对永久性文件系统存储的一致访问权限。

如果另一个用户或进程已经在使用 900（不太可能的情形），操作系统会将 webapp 用户 uid 和 gid 设为另一个默认值。在您的 EC2 实例上运行 Linux 命令 id webapp，以验证分配给 webapp 用户的 uid 和 gid 值。

容错能力

一般来说，设计云架构时，应当考虑那些不令人乐观的情况。利用它提供的弹性。架构的设计、实施和部署目的始终只有一个：即能够自动从故障中恢复。针对 Amazon EC2 实例和 Amazon RDS，使用多可用区域。在概念上，可用区类似于逻辑数据中心。使用 Amazon CloudWatch 更加清楚地了解 Elastic Beanstalk 应用程序的运行状况，以便在出现硬件故障或性能降低的情况下，执行适当操作。配置您的 Auto Scaling 设置，将您的 Amazon EC2 实例组合维持在固定大小，以便使用新的 Amazon EC2 实例替换不正常的实例。如果正在使用 Amazon RDS，请随后设置备份保留期，以便 Amazon RDS 执行自动备份。

内容分发

用户与您的网站连接时，他们的请求可能会通过大量个人网络进行路由。因此，用户可能会由于高延迟导致出现低性能。Amazon CloudFront 可使用遍布全球的边缘站点网络分配您的 Web 内容（如图像、视频等），从而帮助改善延迟问题。用户请求会路由到最近的边缘站点，因此，能以最佳的性能传递内容。CloudFront 可与 Amazon S3 无缝配合，以请持久地存储文件的原始最终版本。有关 Amazon CloudFront 的更多信息，请参阅 Amazon CloudFront 开发人员指南。

软件更新和修补

AWS Elastic Beanstalk 定期发布平台更新以提供修复、软件更新和新功能。Elastic Beanstalk 提供了多种选项来处理平台更新。使用托管平台更新，您的环境会在计划的维护时段内自动升级到最新平台版本，而您的应用程序会继续提供服务。在 2019 年 11 月 25 日或以后使用 Elastic Beanstalk 控制台创建的环境中，默认情况下，尽可能启用托管更新。您还可以使用 Elastic Beanstalk 控制台或 EB CLI 手动启动更新。

连接

Elastic Beanstalk 需要能够连接到环境中的实例才能完成部署。当您在 Amazon VPC 内部署 Elastic Beanstalk 应用程序时，启用连接所需的配置取决于您创建的 Amazon VPC 环境的类型：

- 对于单实例环境，不需要额外的配置。这是因为，在这些环境下，Elastic Beanstalk 会为每个 Amazon EC2 实例分配一个公有弹性 IP 地址，使实例能够直接通过互联网通信。

- 对于同时具有公有子网和私有子网的 Amazon VPC 中负载均衡的可扩展环境，您必须执行以下操作：
  - 在公有子网中创建一个负载均衡器，以将来自互联网的入站流量路由到 Amazon EC2 实例。
  - 创建一个网络地址转换 (NAT) 设备，以将来自私有子网中的 Amazon EC2 实例的出站流量路由到互联网。
  - 为私有子网中的 Amazon EC2 实例创建入站和出站路由规则。
  - 如果使用 NAT 实例，请为 NAT 实例和 Amazon EC2 实例配置安全组以启用互联网通信。
- 对于具有一个公有子网的 Amazon VPC 中负载均衡、可扩展的环境，无需额外的配置。这是因为，在此环境中，Amazon EC2 实例配置有公有 IP 地址，使实例能够与互联网通信。

有关配合使用 Elastic Beanstalk 和 Amazon VPC 的更多信息，请参阅将 Elastic Beanstalk 和 Amazon VPC 结合使用。

服务角色、实例配置文件和用户策略

在创建环境时，AWS Elastic Beanstalk 会提示您提供以下 AWS Identity and Access Management（IAM）角色：

- 服务角色：Elastic Beanstalk 将代入服务角色以代表您使用其他 AWS 服务 服务。
- 实例配置文件：Elastic Beanstalk 将实例配置文件应用于环境中的实例。这将允许实例执行以下操作：
  - 从 Amazon Simple Storage Service（Amazon S3）检索应用程序版本。
  - 将日志上载到 Amazon S3。
  - 执行因环境类型和平台而异的其他任务。

服务角色

在 Elastic Beanstalk 控制台中或使用 Elastic Beanstalk EB CLI 创建环境时，系统会创建所需的服务角色并分配托管式策略。这些策略将会包含所有必要的权限。现在，假设您的账户中已经存在此服务角色，然后在 Elastic Beanstalk 控制台中或使用 Elastic Beanstalk CLI 创建一个新环境。这时，现有的服务角色将会自动分配给新环境。

实例配置文件

如果您的 AWS 账户没有 EC2 实例配置文件，则必须使用 IAM 服务创建一个。然后，您可以将此 EC2 实例配置文件分配到您创建的新环境。创建环境向导提供了相关的信息，以在您使用 IAM 服务过程提供指导，确保您可以创建具有所需权限的 EC2 实例配置文件。创建该实例配置文件后，您可以返回控制台以将其选中并作为 EC2 实例配置文件，然后继续执行创建环境的相关步骤。

> **Note**
>
> 以前，Elastic Beanstalk 会在 AWS 账户首次创建环境时创建一个名为 `aws-elasticbeanstalk-ec2-role` 的默认 EC2 实例配置文件。该实例配置文件包含默认的托管式策略。如果您的账户已经有该实例配置文件，则可继续将其分配到您的环境。
> 但根据最新的 AWS 安全准则，不允许 AWS 服务自动创建具有其他 AWS 服务（在本例中为 EC2）的信任策略的角色。根据这些安全准则，Elastic Beanstalk 将不再创建默认的 `aws-elasticbeanstalk-ec2-role` 实例配置文件。

用户策略

除了您分配给环境的角色外，您还可以创建用户策略并将其应用于账户中的 IAM 用户和组。应用用户策略将会允许用户创建和管理 Elastic Beanstalk 应用程序和环境。Elastic Beanstalk 还提供具有完全访问权限和只读访问权限的托管式策略。有关这些策略的更多信息，请参阅 the section called “用户策略”。

其他实例配置文件和用户策略

您可为高级方案自行创建实例配置文件和用户策略。如果实例需要访问原定设置策略中未包含的服务，您可以创建新策略或者为原定设置策略添加额外的策略。如果托管式策略对于您的需求而言过于宽松，则也可创建更严格的用户策略。有关 AWS 权限的更多信息，请参阅 IAM 用户指南。

主题

- Elastic Beanstalk 服务角色
- Elastic Beanstalk 实例配置文件
- Elastic Beanstalk 用户策略

Elastic Beanstalk 服务角色

服务角色是 Elastic Beanstalk 在代表您调用其他服务时代入的 IAM 角色。例如，Elastic Beanstalk 在调用 Amazon Elastic Compute Cloud（Amazon EC2）、Elastic Load Balancing 和 Amazon EC2 Auto Scaling API 来收集信息时，将会使用一个服务角色。Elastic Beanstalk 使用的服务角色是您在创建 Elastic Beanstalk 环境时指定的角色。

有两个托管策略附加到服务角色。这些策略提供的权限允许 Elastic Beanstalk 访问创建和管理环境所需的 AWS 资源。一个托管策略提供增强的运行状况监控和工作线程层 Amazon SQS 支持的权限，另一个策略提供托管平台更新所需的其他权限。

AWSElasticBeanstalkEnhancedHealth

此策略授予 Elastic Beanstalk 监控环境运行状况所需的所有权限。它还包括 Amazon SQS 操作，以允许 Elastic Beanstalk 监控工作线程环境的队列活动。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeInstanceHealth",
```

```
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetHealth",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:GetConsoleOutput",
                "ec2:AssociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeSecurityGroups",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeNotificationConfigurations",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy

此策略向 Elastic Beanstalk 授予权限，以代表您更新环境以执行托管平台更新。

服务级别权限分组

此策略根据提供的权限集分为多个语句。

- *ElasticBeanstalkPermissions* – 这一组权限用于调用 Elastic Beanstalk 服务操作（Elastic Beanstalk API）。
- *AllowPassRoleToElasticBeanstalkAndDownstreamServices* – 这一组权限允许将任何角色传递给 Elastic Beanstalk 及其他下游服务，例如 AWS CloudFormation。
- *ReadOnlyPermissions* – 这一组权限用于收集有关运行环境的信息。
- **OperationPermissions* – 采用此命名模式的组用于调用执行平台更新必需的操作。
- **BroadOperationPermissions* – 采用此命名模式的组用于调用执行平台更新必需的操作。它们还包括支持旧环境的广泛权限。
- **TagResource* – 采用这种命名模式的组适用于使用 tag-on-create API 为在 Elastic Beanstalk 环境中创建的资源附加标签的调用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ElasticBeanstalkPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticbeanstalk:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowPassRoleToElasticBeanstalkAndDownstreamServices",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "elasticbeanstalk.amazonaws.com",
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "autoscaling.amazonaws.com",
                        "elasticloadbalancing.amazonaws.com",
                        "ecs.amazonaws.com",
                        "cloudformation.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "ReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeLoadBalancers",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeScheduledActions",
                "ec2:DescribeAccountAttributes",
```

```
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstances",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "logs:DescribeLogGroups",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeOrderableDBInstanceOptions",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2BroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
```

```
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2RunInstancesOperationPermissions",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "ec2:LaunchTemplate": "arn:aws:ec2:*:*:launch-template/*"
                }
            }
        },
        {
            "Sid": "EC2TerminateInstancesOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-id": [
                        "arn:aws:cloudformation:*:*:stack/awseb-e-*",
                        "arn:aws:cloudformation:*:*:stack/eb-*"
                    ]
                }
            }
        },
        {
            "Sid": "ECSBroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:DescribeClusters",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ECSDeleteClusterOperationPermissions",
```

```
            "Effect": "Allow",
            "Action": "ecs:DeleteCluster",
            "Resource": "arn:aws:ecs:*:*:cluster/awseb-*"
        },
        {
            "Sid": "ASGOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:AttachInstances",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteScheduledAction",
                "autoscaling:DetachInstances",
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
                "autoscaling:PutScheduledUpdateGroupAction",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:ResumeProcesses",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:SuspendProcesses",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": [
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
awseb-e-*",
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
eb-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/awseb-
e-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/eb-*"
            ]
        },
        {
            "Sid": "CFNOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
```

```
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        },
        {
            "Sid": "ELBOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:targetgroup/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/awseb-*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/eb-*/*"
            ]
        },
        {
            "Sid": "CWLogsOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk/*"
        },
        {
            "Sid": "S3ObjectOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
```

```
                "s3:GetObjectVersionAcl",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectVersionAcl"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*/*"
        },
        {
            "Sid": "S3BucketOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucket",
                "s3:PutBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*"
        },
        {
            "Sid": "SNSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "sns:Subscribe"
            ],
            "Resource": "arn:aws:sns:*:*:ElasticBeanstalkNotifications-*"
        },
        {
            "Sid": "SQSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ],
            "Resource": [
                "arn:aws:sqs:*:*:awseb-e-*",
                "arn:aws:sqs:*:*:eb-*"
            ]
        },
        {
            "Sid": "CWPutMetricAlarmOperationPermissions",
            "Effect": "Allow",
```

```
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:awseb-*",
                "arn:aws:cloudwatch:*:*:alarm:eb-*"
            ]
        },
        {
            "Sid": "AllowECSTagResource",
            "Effect": "Allow",
            "Action": [
                "ecs:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterTaskDefinition"
                    ]
                }
            }
        }
    ]
}
```

您可以使用以下任一方法创建 Elastic Beanstalk 环境。每个部分都描述了该方法如何处理服务角色。

Elastic Beanstalk 控制台

使用 Elastic Beanstalk 控制台创建环境时，Elastic Beanstalk 将提示您创建一个名为 `aws-elasticbeanstalk-service-role` 的服务角色。当通过 Elastic Beanstalk 创建时，此角色包括一个信任策略，允许 Elastic Beanstalk 代入服务角色。本主题前面描述的两个托管策略也会附加到该角色。

Elastic Beanstalk 命令行界面 (EB CLI)

您可以使用 Elastic Beanstalk 命令行界面（EB CLI）的 the section called “eb create” 命令创建环境。如果您未通过 `--service-role` 选项指定服务角色。Elastic Beanstalk 将创建相同的默认服务角色 `aws-elasticbeanstalk-service-role`。如果默认服务角色已存在，Elastic Beanstalk 会将其用于新环境。当通过 Elastic Beanstalk 创建时，此角色包括一个信任策略，允许 Elastic Beanstalk 代入服务角色。本主题前面描述的两个托管策略也会附加到该角色。

Elastic Beanstalk API

您可以使用 Elastic Beanstalk API 的 `CreateEnvironment` 操作创建环境。如果您未指定服务角色，Elastic Beanstalk 将创建一个监控服务相关角色。这是独特类型的服务角色，由 Elastic Beanstalk 预定义，它具有服务代表您调用其他 AWS 服务所需的所有权限。服务相关角色与您的账户关联。Elastic Beanstalk 仅会创建此角色一次，然后在创建其他环境时重复使用此角色。您也可以使用 IAM 提前为账户创建此监控服务相关角色。在您的账户具有监控服务相关角色时，您可以通过 Elastic Beanstalk 控制台、Elastic Beanstalk API 或 EB CLI 使用该角色创建环境。有关如何将服务相关角色与 Elastic Beanstalk 环境结合使用的说明，请参阅将服务相关角色用于 Elastic Beanstalk。

有关服务角色的更多信息，请参阅管理 Elastic Beanstalk 服务角色。

Elastic Beanstalk 实例配置文件

实例配置文件是一种 IAM 角色，将应用到在您的 Elastic Beanstalk 环境中启动的 Amazon EC2 实例。在创建 Elastic Beanstalk 环境时，您可以指定 EC2 实例执行下列操作时将使用的实例配置文件：

- 从 Amazon Simple Storage Service (Amazon S3) 检索 应用程序版本
- 将日志写入 Amazon S3
- 在 AWS X-Ray 集成环境中，向 X-Ray 上传调试数据
- 在 Amazon ECS 托管式 Docker 环境中，使用 Amazon Elastic Container Service（Amazon ECS）协调容器部署
- 在工作线程环境中，从 Amazon Simple Queue Service (Amazon SQS) 队列读取
- 在工作线程环境下：使用 Amazon DynamoDB 进行领导选择
- 在工作线程环境下，将实例运行状况指标发布到 Amazon CloudWatch

Elastic Beanstalk 提供了一组托管式策略，以便您环境中的 EC2 实例执行所需的操作。基本使用场景所需的托管式样策略如下。

- `AWSElasticBeanstalkWebTier`
- `AWSElasticBeanstalkWorkerTier`
- `AWSElasticBeanstalkMulticontainerDocker`

在 Elastic Beanstalk 控制台中首次启动环境时，您需要将这些策略附加到您创建的实例配置文件。

如果 Web 应用程序需要访问其他 AWS 服务，请向该实例配置文件添加允许访问这些服务的语句或托管式策略。

有关实例配置文件的更多信息，请参阅管理 Elastic Beanstalk 实例配置文件。

Elastic Beanstalk 用户策略

为使用 Elastic Beanstalk 的每个用户创建 IAM 用户，以避免使用根账户或者共享凭证。作为一项安全性最佳实践，请只向这些用户授予访问所需服务和功能的权限。

Elastic Beanstalk 不仅需要其自身的 API 操作的权限，还需要多个其他 AWS 服务的权限。Elastic Beanstalk 使用用户权限在环境中启动资源。这些资源包括若干 EC2 实例、一个 Elastic Load Balancing 负载均衡器和一个 Auto Scaling 组。Elastic Beanstalk 还使用用户权限将日志和模板保存到 Amazon Simple Storage Service (Amazon S3)、向 Amazon SNS 发送通知、分配实例配置文件以及向 CloudWatch 发布指标。Elastic Beanstalk 需要 AWS CloudFormation 权限以协调资源部署和更新。它还需要 Amazon RDS 权限以根据需要创建数据库，需要 Amazon SQS 权限为工作线程环境创建队列。

有关用户策略的详细信息，请参阅管理 Elastic Beanstalk 用户策略。

Elastic Beanstalk 平台

AWS Elastic Beanstalk 提供了多种平台，您可以在这些平台上构建应用程序。您可以根据这些平台之一设计 Web 应用程序，Elastic Beanstalk 将代码部署到您选择的平台版本以创建有效的应用程序环境。

Elastic Beanstalk 为不同的编程语言、应用程序服务器以及 Docker 容器提供平台。某些平台具有多个同时支持的版本。

主题

Elastic Beanstalk 平台词汇表

以下是与 AWS Elastic Beanstalk 平台及其生命周期相关的关键术语。

运行时

运行您的应用程序代码所需的编程语言特定的运行时软件（框架、库、解释器、虚拟机等）。

Elastic Beanstalk 组件

Elastic Beanstalk 添加到平台以启用 Elastic Beanstalk 功能的软件组件。例如，增强型运行状况代理是收集和报告运行状况信息所必需的。

平台

操作系统 (OS)、运行时、Web 服务器、应用程序服务器和 Elastic Beanstalk 组件的组合。平台提供可用于运行您的应用程序的组件。

Platform Version

操作系统 (OS)、运行时、Web 服务器、应用程序服务器和 Elastic Beanstalk 组件的特定版本的组合。您可以基于平台版本创建 Elastic Beanstalk 环境，然后将应用程序部署到该环境中。

平台版本具有格式为 X.Y.Z 的语义版本号，其中，X 是主要版本，Y 是次要版本，Z 是补丁版本。

平台版本可以处于下列状态之一：

- 支持 – 完全由支持的组件 构成的平台版本。所有组件尚未达到其各自供应商（拥有者：AWS或第三方；或社区）指定的使用寿命结束 (EOL) 日期。它们会定期收到供应商提供的补丁或次要更新。Elastic Beanstalk 使受支持的平台版本可供您用于创建环境。
- 已停用 – 具有一个或多个已停用组件（这些组件已达到其供应商指定的使用寿命结束 (EOL) 日期）的平台版本。停用的平台版本无法供新客户或现有客户用于 Elastic Beanstalk 环境中。

  有关已停用组件的详细信息，请参阅the section called “平台支持策略”。

平台分支

一系列平台版本，它们共用某些组件的特定版本（通常是主要版本），例如操作系统 (OS)、运行时或 Elastic Beanstalk 组件。例如：64 位 Amazon Linux 上运行的 Python 3.6；64 位 Windows Server 2016 上运行的 IIS 10.0。分支中的每个连续平台版本均是对上一个版本的更新。

可以无条件地使用每个平台分支中的最新平台版本来创建环境。分支中之前的平台版本仍受支持 – 您可以根据先前的平台版本创建环境（如果您在过去的 30 天内在环境中使用了先前的平台版本）。但是，这些先前的平台版本缺乏最新的组件，因此不推荐使用。

平台分支可以处于下列状态之一：

- 支持 – 当前平台分支。它完全由支持的组件构成。它接收持续的平台更新，推荐将其用于生产环境。有关支持的平台分支的列表，请参阅 AWS Elastic Beanstalk 平台指南中的 Elastic Beanstalk 支持的平台。

- 测试版 – 预发布的预览版平台分支。它是实验性的。它可能会在一段时间内接收持续的平台更新，但没有长期支持。建议不要在生产环境中使用测试版平台分支。仅将它用于评估。有关测试版平台分支的列表，请参阅 AWS Elastic Beanstalk 平台指南中的 公开测试版中的 Elastic Beanstalk 平台版本。
- 弃用 – 具有一个或多个弃用组件 的平台分支。它接收持续的平台更新，但建议不要将它用于生产环境。有关弃用的平台分支的列表，请参阅 AWS Elastic Beanstalk 平台指南中的计划停用的 Elastic Beanstalk 平台版本。
- 停用 – 具有一个或多个停用组件 的平台分支。它不再接收平台更新，建议不要将它用于生产环境。AWS Elastic Beanstalk 平台指南中未列出停用的平台分支。Elastic Beanstalk 不会使已停用的平台分支的平台版本可供您用于创建环境。

支持的组件 没有其供应商（拥有者或社区）计划的停用日期。供应商可能是AWS或第三方。弃用的组件 具有其供应商计划的停用日期。停用的组件 已达到使用寿命终止 (EOL) 日期，并且不再受其供应商的支持。有关已停用组件的详细信息，请参阅the section called “平台支持策略”。

如果您的环境使用弃用或停用的平台分支，我们建议您将其更新为受支持的平台分支中的平台版本。有关详细信息，请参阅 the section called “平台更新”。

平台更新

发布新的平台版本，其中包含对于平台的一些组件（操作系统、运行时、Web 服务器、应用程序服务器和 Elastic Beanstalk 组件）的更新。平台更新遵循语义版本分类，并且可能具有多个级别：

- 主要更新 – 具有与现有平台版本不兼容的更改的更新。您可能需要修改您的应用程序以便在新的主要版本上正常运行。主要更新具有新的主要平台版本号。
- 次要更新 – 添加向后兼容现有平台版本的功能的更新。您不需要修改您的应用程序，即可在新的次要版本上正常运行。次要更新具有新的次要平台版本号。
- 补丁更新 – 由向后兼容现有平台版本的维护版本（错误修复、安全更新和性能改进）组成的更新。补丁更新具有新的补丁平台版本号。

托管更新

一项 Elastic Beanstalk 功能，该功能自动对 Elastic Beanstalk 支持的平台版本的操作系统 (OS)、运行时、Web 服务器、应用程序服务器和 Elastic Beanstalk 组件应用补丁和次要更新。托管更新将同一平台分支中的较新平台版本应用于您的环境。您可以配置托管平台更新以仅应用补丁更新、次要版本更新，或同时应用这两种更新。您还可以完全禁用托管更新。

有关更多信息，请参阅 托管平台更新。

Elastic Beanstalk 平台维护的责任共担模型

AWS 而且，我们的客户有责任实现高水平的软件组件安全性和合规性。此共享模型可以降低您的运营负担。

有关详细信息，请参阅责任 AWS 共担模型。

AWS Elastic Beanstalk 通过提供托管更新功能，帮助您履行责任共担模式。此功能会自动为 Elastic Beanstalk 支持的平台版本应用补丁更新和次要版本更新。如果托管更新失败，Elastic Beanstalk 会通知您出现故障，以确保您发现故障并可以立即采取措施。

有关更多信息，请参阅 托管平台更新。

此外，Elastic Beanstalk 还执行以下操作：

- 发布其未来 12 个月的平台支持策略和停用计划。
- 发布操作系统 (OS)、运行时、应用程序服务器和 Web 服务器组件的补丁更新、次要更新和主要更新（通常在这些更新推出后的 30 天内）。Elastic Beanstalk 负责创建对其支持的平台版本上现有的 Elastic Beanstalk 组件的更新。所有其他更新都直接来自其供应商（所有者或社区）。

我们在 AWS Elastic Beanstalk 发行说明指南中的发行说明中公布了我们支持的平台的所有更新。我们还在 AWS Elastic Beanstalk 平台指南中提供了所有支持的平台及其组件的列表以及平台历史记录。有关更多信息，请参阅 支持的平台。

您负责执行以下操作：

- 更新您控制的所有组件（在责任 AWS 共担模型中标识为客户）。这包括确保您的应用程序、数据以及应用程序所需和您下载的任何组件的安全性。
- 确保您的 Elastic Beanstalk 环境在任何受支持的平台版本上运行，并将在停用的平台版本上运行的任何环境迁移到受支持的版本。
- 解决失败的托管更新尝试中遇到的所有问题，然后重试更新。
- 如果您选择退出 Elastic Beanstalk 托管更新，则需自行修补操作系统、运行时、应用程序服务器和 Web 服务器。为此，您可以手动应用平台更新或直接修补所有相关环境资源中的组件。
- 根据 AWS 责任共担模型，管理您在 Elastic Beanstalk 之外使用的任何 AWS 服务的安全性和合规性。

Elastic Beanstalk 平台支持策略

AWS Elastic Beanstalk 提供了用于在上面运行应用程序的各种平台 AWS。Elastic Beanstalk 支持仍从其供应商（拥有者或社区）接收持续次要版本更新和补丁更新的平台分支。有关相关术语的完整定义，请参阅 Elastic Beanstalk 平台词汇表。

停用的平台分支

当受支持平台分支的某个组件被其供应商标记为使用寿命结束 (EOL) 时，Elastic Beanstalk 会将该平台分支标记为已停用。平台分支的组件包括：操作系统 (OS)、运行时语言版本、应用程序服务器或 Web 服务器。

平台分支被标记为已停用后，将适用以下政策：

- Elastic Beanstalk 停止提供维护更新，包括安全更新。
- Elastic Beanstalk 不再为已停用的平台分支机构提供技术支持。
- Elastic Beanstalk 不再向新的 Elastic Beanstalk 客户提供平台分支以部署到新环境。对于在已停用平台分支上运行的活动环境的现有客户，自公布的停用日期起有 90 天的宽限期。

> **Note**
>
> 已停用的平台分支将无法在 Elastic Beanstalk 控制台中使用。但是，对于拥有基于已停用平台分支的现有环境的客户，它将通过 EB CLI 和 EB API 提供。 AWS CLI现有客户还可以使用克隆环境和重建环境控制台。

有关计划停用的平台分支的列表，请参阅接下来的 Elastic Beanstalk 平台计划主题停用平台分支计划中的。

有关环境的平台分支停用后预期情况的更多信息，请参阅平台停用常见问题。

超过 90 天宽限期

我们针对已停用平台分支的政策不会删除对环境的访问权限，也不会删除资源。但是，在已停用的平台分支上运行 Elastic Beanstalk 环境的现有客户应该意识到这样做的风险。这样的环境最终可能会陷入不可预测的境地，因为 Elastic Beanstalk 无法为已停用的平台分支提供安全更新、技术支持或修补程序，因为供应商将其组件标记为已停产。

例如，在已停用平台分支上运行的环境中，可能会出现有害且关键的安全漏洞。或者，如果环境随着时间推移变得与 Elastic Beanstalk 服务不兼容，则 EB API 操作可能会不再适用于该环境。已停用平台分支上的环境保持活动状态的时间越长，出现这些类型风险的几率就越高。要继续受益于组件供应商在更新的版本中提供的重要安全性、性能和功能增强，我们强烈建议您将所有 Elastic Beanstalk 环境更新为支持的平台版本。

如果您的应用程序在停用的平台分支上运行时遇到问题，并且您无法将其迁移到支持的平台，则需要考虑其他替代方案。解决方法包括将该应用程序封装到 Docker 映像中，以便将其以 Docker 容器的形式运行。这将允许客户使用我们的任何 Docker 解决方案，例如我们的 Elastic Beanstalk AL2023/AL2 Docker 平台，或者其他基于 Docker 的服务，例如亚马逊 ECS 或 Amazon EKS。非 Docker 替代方案包括我们的 AWS CodeDeploy 服务，它允许对你想要的运行时进行完全自定义。

Elastic Beanstalk 平台发布时间表

为了确保您的应用程序在受支持和安全的环境中运行，Elastic Beanstalk 会定期为其托管平台提供更新，如上一主题所述。责任共担模式除了每月发布新的平台分支版本外，我们的版本维护还包括以下流程：

- 发布新的平台分支 — 这些分支通常会引入运行时语言、操作系统或应用程序服务器的新主要版本。
- 平台分支的停用 — 当平台分支的一个组件达到使用寿命终止 (EOL) 时，我们必须将其停用。有关我们退休分支机构政策的更多信息，请参阅 Elastic Beanstalk 平台支持策略

主题

规划资源

除了随后的时间表外，还有其他资源可以帮助您为在 Elastic Beanstalk 平台上运行的应用程序规划维护和支持。有关我们的平台组件、重要日期和发布公告的更多信息，请参阅以下资源：

- AWS Elastic Beanstalk 平台指南 — 本指南提供了我们每个平台分支的详细组件列表。它还按发布日期提供平台历史记录，其中包含相同的详细信息。当您的平台分支的特定组件发生变化时，本指南可

以通知您。如果您的应用程序开始表现不同，您还可以在平台指南中交叉引用发生日期，以查看是否有任何平台更改可能影响了您的应用程序。

- AWS Elastic Beanstalk 发行说明 — 我们的发行说明宣布了我们所有的平台版本，包括次要版本和主要版本。这包括我们的每月平台更新、安全发布、修补程序和停用公告。您可以从发行说明文档中订阅我们的 RSS feed。

即将发布的平台分支版本

下表列出了即将推出的 Elastic Beanstalk 平台分支及其目标发布日期。这些日期是暂定的，可能会发生变化。

运行时版本/平台分支	操作系统	目标发布日期
Corretto 21 with Tomcat 10 AL2023	Amazon Linux 2023	2024 年 9 月
PHP 8.3 AL2023	Amazon Linux 2023	2024 年 9 月
Python 3.12 AL2023	Amazon Linux 2023	2024 年 9 月
Ruby 3.3 AL2023	Amazon Linux 2023	2024 年 11 月

停用平台分支计划

下表列出了计划停用的 Elastic Beanstalk 平台分支，因为它们的某些组件已接近使用寿命 (EOL)。

有关包括其特定组件的停用平台分支的更详细列表，请参阅平台指南中的停用AWS Elastic Beanstalk平台版本。

运行时版本/平台分支	操作系统	目标退休日期
Corretto 8 with Tomcat 8.5 AL2	Amazon Linux 2	2024 年 9 月 30 日
Corretto 11 with Tomcat 8.5 AL2	Amazon Linux 2	2024 年 9 月 30 日

运行时版本/平台分支	操作系统	目标退休日期
.NET 6 AL2023	Amazon Linux 2023	2025 年 1 月 31 日
Node.js 14 AL2	Amazon Linux 2	2024 年 9 月 30 日
Node.js 16 AL2	Amazon Linux 2	2024 年 9 月 30 日
Ruby 2.7 AL2	Amazon Linux 2	2024 年 9 月 30 日
Ruby 3.0 AL2	Amazon Linux 2	2024 年 9 月 30 日
PHP 8.0 AL2	Amazon Linux 2	2024 年 9 月 30 日
PHP 8.1 AL2	Amazon Linux 2	2025 年 1 月 31 日
PHP 8.1 AL2023	Amazon Linux 2023	2025 年 1 月 31 日
Python 3.7 AL2	Amazon Linux 2	2024 年 9 月 30 日
Python 3.8 AL2	Amazon Linux 2	2025 年 1 月 31 日

已停用平台分支历史记录

下表列出了已经处于停用状态的 Elastic Beanstalk 平台分支。您可以在《平台历史记录》指南中查看这些平台分支及其组件的AWS Elastic Beanstalk 详细历史记录。

Amazon Linux 2 (AL2)

运行时版本/平台分支	停用日期		
Corretto 11 with Tomcat 7 AL2	2022 年 6 月 29 日		
Corretto 8 with Tomcat 7 AL2	2022 年 6 月 29 日		
Node.js 12 AL2	2022 年 12 月 23 日		

运行时版本/平台分支	停用日期		
Node.js 10 AL2	2022 年 6 月 29 日		
PHP 7.4 AL2	2023 年 6 月 9 日		
PHP 7.3 AL2	2022 年 6 月 29 日		
PHP 7.2 AL2	2022 年 6 月 29 日		
Ruby 2.6 AL2	2022 年 12 月 23 日		
Ruby 2.5 AL2	2022 年 6 月 29 日		

Amazon Linux AMI（AL1）

运行时版本/平台分支	停用日期		
Single Container Docker	2022 年 7 月 18 日		
Multicontainer Docker	2022 年 7 月 18 日		
Preconfigured Docker - GlassFish 5.0 with Java 8	2022 年 7 月 18 日		
Go 1	2022 年 7 月 18 日		
Java 8	2022 年 7 月 18 日		
Java 7	2022 年 7 月 18 日		
Java 8 with Tomcat 8.5	2022 年 7 月 18 日		

运行时版本/平台分支	停用日期		
Java 7 with Tomcat 7	2022 年 7 月 18 日		
Node.js	2022 年 7 月 18 日		
PHP 7.2 - 7.3	2022 年 7 月 18 日		
Python 3.6	2022 年 7 月 18 日		
Ruby 2,4, 2.5, 2.6 with Passenger	2022 年 7 月 18 日		
Ruby 2.4, 2.5, 2.6 with Puma	2022 年 7 月 18 日		
Go 1.3–1.10	2020 年 10 月 31 日		
Java 6	2020 年 10 月 31 日		
Node.js 4.x–8.x	2020 年 10 月 31 日		
PHP 5.4–5.6	2020 年 10 月 31 日		
PHP 7.0–7.1	2020 年 10 月 31 日		
Python 2.6、2.7、3.4	2020 年 10 月 31 日		
Ruby 1.9.3	2020 年 10 月 31 日		
Ruby 2.0–2.3	2020 年 10 月 31 日		

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关更多信息，请参阅 平台停用常见问题。

Windows Server

运行时版本/平台分支	停用日期		
在 64 位 Windows Server（和 Core）2012 R2 版本 0.1.0 上运行的 IIS 8.5	2022 年 6 月 29 日		
在 64 位 Windows Server（和 Core）2012 R2 版本 1.2.0 上运行的 IIS 8.5	2022 年 6 月 29 日		
在 64 位 Windows Server 2016（和 Core）版本 1.2.0 上运行的 IIS 10.0	2022 年 6 月 29 日		
在 64 位 Windows Server 2012 R1 平台分支上运行的 IIS 8	2022 年 6 月 22 日		
在 64 位 Windows Server 2012 R1 版本	2022 年 6 月 22 日		

运行时版本/平台分支	停用日期		
0.1.0 上运行的 IIS 8			
在 64 位 Windows Server 2012 R1 版本 1.2.0 上运行的 IIS 8	2022 年 6 月 22 日		

> **Note**
>
> 有关 Windows 2012 R2 平台分支停用的更多信息，请参阅《AWS Elastic Beanstalk 发行说明》中的 Windows Server 2012 R2 平台分支已停用。

已停用的服务器和操作系统历史记录

下表提供了 Elastic Beanstalk 平台不再支持的操作系统、应用程序服务器和 Web 服务器的历史记录。使用这些组件的所有平台分支现已停用。这些日期反映了包含该组件的最后一个 Elastic Beanstalk 平台分支的停用日期。

操作系统

操作系统版本	平台停用日期		
Windows Server 2012 R2 running IIS 8.5	2023 年 12 月 4 日		
Windows Server Core 2012 R2 running IIS 8.5	2023 年 12 月 4 日		
Amazon Linux AMI（AL1）	2022 年 7 月 18 日		

操作系统版本	平台停用日期		
Windows Server 2012 R1	2022 年 6 月 22 日		
Windows Server 2008 R2	2019 年 10 月 28 日		

应用程序服务器

应用程序服务器版本	平台停用日期		
Tomcat 7	2022 年 6 月 29 日，针对 Amazon Linux 2（AL2）平台 2022 年 7 月 18 日，针对 Amazon Linux AMI（AL1）平台		
Tomcat 8	2020 年 10 月 31 日		
Tomcat 6	2020 年 10 月 31 日		

Web 服务器

Web 服务器版本	平台停用日期		
在 64 位 Windows Server 上运行的 IIS 8	2022 年 6 月 22 日		
Apache HTTP 服务器 2.2	2020 年 10 月 31 日		
Nginx 1.12.2	2020 年 10 月 31 日		

Elastic Beanstalk 支持的平台

AWS Elastic Beanstalk 提供了多种平台，您可以在这些平台上构建应用程序。您可以根据这些平台之一设计 Web 应用程序，Elastic Beanstalk 将代码部署到您选择的平台版本以创建有效的应用程序环境。

Elastic Beanstalk 为编程语言（Go、Java、Node.js、PHP、Python、Ruby）、应用程序服务器（Tomcat、Passenger、Puma）和 Docker 容器提供平台。某些平台具有多个同时支持的版本。

Elastic Beanstalk 会预配置运行您的应用程序所需的资源，包括一个或多个 Amazon EC2 实例。Amazon EC2 实例上运行的软件堆栈取决于您为您的环境选择的特定平台版本。

您可以自定义和配置您的应用程序在平台中所依赖的软件。有关更多信息，请参阅自定义 Linux 服务器上的软件和自定义 Windows Server 上的软件。有关最新版本的详细发布说明，请参阅 AWS Elastic Beanstalk 发布说明。

支持的平台

AWS Elastic Beanstalk 平台指南在 Elasti c Beanstalk 支持的平台部分列出了所有当前平台分支版本。平台指南还列出了每个平台的平台历史记录，其中包括以前的分支平台版本列表。要查看每个平台的平台历史记录，请选择以下链接之一。

- Docker
- Go
- Java SE
- Tomcat（运行 Java SE）
- Linux 上的 .NET Core
- Windows Server 上的 .NET
- Node.js
- PHP
- Python
- Ruby

平台分支的解决方案堆栈名称

您可以使用给定平台分支版本的解决方案堆栈名称通过 EB CLI、El astic Beanstalk API 或 CL I 启动环境。AWSAWS Elastic Beanstalk 平台指南在 Elasti c Beanstalk 支持的平台和平台历史记录部分中列出了平台分支版本下的解决方案堆栈名称。

要检索可用于创建环境的所有解决方案堆栈名称，请使用 ListAvailableSolutionStacksAPI 或 AWS CLI `aws elasticbeanstalk list-available-solution-stacks`中的。

Elastic Beanstalk Linux 平台

Elastic Beanstalk 支持的大多数平台都基于 Linux 操作系统。具体而言，这些平台基于亚马逊 Linux（由提供的 Linux 发行版） AWS。Elastic Beanstalk Linux 平台使用 Amazon Elastic Compute Cloud（Amazon EC2）实例，这些实例运行 Amazon Linux。

Elastic Beanstalk Linux 平台提供了许多开箱即用的功能。您可以通过多种方式扩展平台以支持您的应用程序。有关更多信息，请参阅 the section called “扩展 Linux 平台”。

主题

- 受支持的 Amazon Linux 版本
- Elastic Beanstalk Linux 平台列表
- 扩展 Elastic Beanstalk Linux 平台

受支持的 Amazon Linux 版本

AWS Elastic Beanstalk 支持基于亚马逊 Linux 2 和亚马逊 Linux 2023 的平台。

截至 2023 年 10 月 19 日，Elastic Beanstalk 为 Amazon Linux 2 平台也支持的所有编程语言提供 AL2023 平台。Beanstalk 还在 Amazon Linux 2 和 Amazon Linux 2023 上支持 Docker 和基于 ECS 的 Docker 平台。

有关 Amazon Linux 2 和 Amazon Linux 2023 的更多信息，请参阅以下文件：

- 亚马逊 Linux 2 — 亚马逊 EC2 用户指南中的亚马逊 L inux。
- Amazon Linux 2023 – Amazon Linux 2023 用户指南中的什么是 Amazon Linux 2023？。

有关支持的平台版本的更多信息，请参阅 Elastic Beanstalk 支持的平台。

> **Note**
>
> 您可以将应用程序从 Elastic Beanstalk AL1 或 AL2 平台分支迁移到等效的 AL2023 平台分支。有关更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

Amazon Linux 2023

AWS 2023 年 3 月宣布亚马逊 Linux 2023 正式上市。Amazon Linux 2023 用户指南总结了 Amazon Linux 2 与 Amazon Linux 2023 之间的主要差异。有关更多信息，请参阅用户指南中的比较 Amazon Linux 2 和 Amazon Linux 2023。

Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台之间具有高度的兼容性。尽管还有一些差异需要注意：

- 实例元数据服务版本 1（IMDSv1）– DisableIMDSv1 选项设置在 AL2023 平台上默认为 `true`。在 AL2 平台上默认为 `false`。
- pkg-repo 实例工具 – pkg-repo 工具不适用于在 AL2023 平台上运行的环境。但是，您可以手动将软件包和操作系统更新应用于 AL2023 实例。有关更多信息，请参阅 Amazon Linux 2023 用户指南中的管理软件包和操作系统更新。
- Apache HTTPd 配置 – AL2023 平台的 Apache `httpd.conf` 文件中的一些配置设置与 AL2 的配置设置不同：
  - 默认情况下，拒绝访问服务器的整个文件系统。这些设置在 Apache 网站安全提示页面上的默认保护服务器文件中进行了描述。
  - 阻止用户覆盖您配置的安全功能。该配置拒绝访问所有目录中的 `.htaccess` 设置，专门启用的目录除外。此设置在 Apache 网站安全提示页面上的保护系统设置中进行了描述。Apache HTTP 服务器教程：.htaccess 文件页面指出，此设置可能有助于提高性能。
  - 拒绝访问带有名称模式 `.ht*` 的文件。此设置阻止 Web 客户端查看 `.htaccess` 和 `.htpasswd` 文件。

您可以更改您的环境的上述任何配置设置。有关更多信息，请参阅 扩展 Elastic Beanstalk Linux 平台。展开反向代理主题以查看配置 Apache HTTPD 部分。

Elastic Beanstalk Linux 平台列表

下面的列表提供了 Elastic Beanstalk 针对不同编程语言和 Docker 容器支持的 Linux 平台。Elastic Beanstalk 为所有平台提供基于 Amazon Linux 2 和 Amazon Linux 2023 的平台。要了解有关平台的更多信息，请选择对应的链接。

- Docker（和 ECS Docker）
- Go
- Tomcat（运行 Java SE）
- Java SE
- Linux 上的 .NET Core
- Node.js
- PHP
- Python
- Ruby

扩展 Elastic Beanstalk Linux 平台

AWS Elastic Beanstalk Linux 平台提供了许多开箱即用的功能，以支持您的应用程序的开发和运行。您可以根据需要通过多种方式扩展平台，以配置选项、安装软件、添加文件和启动命令、提供构建和运行时说明，以及添加在环境 Amazon Elastic Compute Cloud (Amazon EC2) 实例的不同预配置阶段运行的初始化脚本。

Buildfile 和 Procfile

某些平台允许您自定义如何构建或准备应用程序，并指定运行应用程序的进程。每个平台主题都特别提到 Buildfile 和/或 Procfile（如果平台支持）。在 平台 下查找您的特定平台。

对于所有支持的平台，语法和语义是相同的，如本页所述。各个平台主题都提到了这些文件在以各自的语言构建和运行应用程序时的特定用法。

Buildfile

要为应用程序指定自定义构建和配置命令，请将名为 `Buildfile` 的文件放置在应用程序源的根目录中。文件名区分大小写。`Buildfile` 使用以下语法。

```
<process_name>: <command>
```

Buildfile 中的命令必须符合以下正则表达式：^[A-Za-z0-9_-]+:\s*[^\s].*$

Elastic Beanstalk 不会监控通过 Buildfile 运行的应用程序。对于短期运行并在完成任务后终止的命令，请使用 Buildfile。对于长期运行、不应退出的应用程序进程，请使用 Procfile。

Buildfile 中的所有路径都是源包根目录的相对路径。在下面的 Buildfile 示例中，build.sh 是位于源包根目录的 Shell 脚本。

Example Buildfile

```
make: ./build.sh
```

如果您想提供自定义构建步骤，我们建议您将 predeploy 平台挂钩用于除最简单的命令之外的任何内容，而不是 Buildfile。通过平台挂钩可以使用更丰富的脚本，并且能更好地进行错误处理。平台挂钩将在下一节中介绍。

Procfile

要指定自定义命令以便启动和运行应用程序，请将名为 Procfile 的文件放置在应用程序源的根目录中。文件名区分大小写。Procfile 使用以下语法。您可以指定一个或多个命令。

```
<process_name1>: <command1>
<process_name2>: <command2>
...
```

Procfile 中的每一行都必须符合以下正则表达式：^[A-Za-z0-9_-]+:\s*[^\s].*$

对于长期运行、不应退出的应用程序进程，请使用 Procfile。Elastic Beanstalk 希望从 Procfile 运行的进程一直运行。Elastic Beanstalk 会监控这些进程并重启所有终止的进程。对于短期运行的进程，请使用 Buildfile。

Procfile 中的所有路径都是源包根目录的相对路径。以下示例 Procfile 定义了三个进程。第一个进程在该示例中称为 web，它是主要 Web 应用程序。

Example Procfile

```
web: bin/myserver
cache: bin/mycache
foo: bin/fooapp
```

Elastic Beanstalk 将代理服务器配置为将请求转发到端口 5000 上的主 Web 应用程序，并且您可以配置此端口号。`Procfile` 的常见用途是将此端口号作为命令参数传递给应用程序。有关代理配置的详细信息，请展开此页上的反向代理配置部分。

Elastic Beanstalk 将 `Procfile` 进程的标准输出和错误流捕获到日志文件中。Elastic Beanstalk 根据进程名称命名日志文件并将其存储在 `/var/log` 中。例如，上例中的 `web` 进程分别为 `web-1.log` 和 `web-1.error.log` 生成名为 `stdout` 和 `stderr` 的日志。

平台挂钩

平台挂钩是专为扩展您的环境平台而设计的。这些自定义脚本和其他可执行文件部署为应用程序源代码的一部分，并由 Elastic Beanstalk 在不同的实例预置阶段运行。

> **Note**
>
> Amazon Linux AMI 平台版本（Amazon Linux 2 之前的版本）不支持平台挂钩。

应用程序部署平台挂钩

当您提供新的源包进行部署时，或者当您进行的配置更改要求终止和重新创建所有环境实例时，将发生应用程序部署。

要提供在应用程序部署期间运行的平台挂钩，请将文件放在源包中的 `.platform/hooks` 目录下（位于以下子目录之一中）。

- `prebuild` – 此处的文件在 Elastic Beanstalk 平台引擎下载和提取应用程序源包之后且在设置和配置应用程序和 Web 服务器之前运行。

  `prebuild` 文件在运行任何配置文件的 commands 部分中的命令之后且在运行 `Buildfile` 命令之前运行。
- `predeploy` – 此处的文件在 Elastic Beanstalk 平台引擎设置和配置应用程序及 Web 服务器之后且在将它们部署到其最终运行时位置之前运行。

  `predeploy` 文件在运行任何配置文件的 container_commands 部分中的命令之后且在运行 `Procfile` 命令之前运行。
- `postdeploy` – 此处的文件在 Elastic Beanstalk 平台引擎部署应用程序和代理服务器之后运行。

  这是最后一个部署工作流步骤。

配置部署平台挂钩

如果您进行的配置更改仅更新环境实例而不重新创建环境实例，则会发生配置部署。以下选项更新会导致配置更新。

- 环境属性和特定于平台的设置
- 静态文件
- AWS X-Ray 守护程序
- 日志存储和流式处理
- 应用程序端口（有关详细信息，请展开此页上的反向代理配置部分）

要提供在配置部署期间运行的挂钩，请将它们放在源包中的 `.platform/confighooks` 目录下。应用与应用程序部署挂钩相同的三个子目录。

有关平台挂钩的更多信息

挂钩文件可以是二进制文件，也可以是以包含其解释器路径的 `#!` 行开头的脚本文件，例如 `#!/bin/bash`。所有文件都必须具有执行权限。使用 `chmod +x` 对挂钩文件设置执行权限。对于 2022 年 4 月 29 日或之后发布的所有基于 Amazon Linux 2023 和 Amazon Linux 2 的平台版本，Elastic Beanstalk 会自动向所有平台挂钩脚本授予执行权限。在这种情况下，您无需手动授予执行权限。有关这些平台版本的列表，请参阅《AWS Elastic Beanstalk 发布说明指南》中的 2022 年 4 月 29 日 Linux 发布说明。

Elastic Beanstalk 按照文件名的字母表顺序运行上述每个目录中的文件。所有文件都以 `root` 用户身份运行。平台挂钩的当前工作目录 (cwd) 是应用程序的根目录。对于 `prebuild` 和 `predeploy` 文件，该目录是应用程序暂存目录，对于 `postdeploy` 文件，该目录是当前应用程序目录。如果其中一个文件失败（以非零退出代码退出），则部署中止并失败。

如果平台挂钩文本脚本包含 Windows 回车符/换行符 (CRLF) 换行符，则该脚本可能会失败。如果某文件保存在 Windows 主机中，然后传输到 Linux 服务器，则它可能包含 Windows CRLF 换行符。对于 2022 年 12 月 29 日当天或之后发布的平台，Elastic Beanstalk 会自动将 Windows CRLF 字符转换为平台挂钩文本文件中的 Linux 换行符 (LF) 换行符。如果应用程序在此日期之前发布的任何 Amazon Linux 2 平台上运行，则需要将 Windows CRLF 字符转换为 Linux LF 字符。实现此目的的一种方法是在 Linux 主机上创建并保存脚本文件。互联网上也提供了转换这些字符的工具。

挂钩文件可以访问您在应用程序选项中定义的所有环境属性和系统环境变量 `HOME`、`PATH` 和 `PORT`。

要将环境变量和其他配置选项的值添加到平台挂钩脚本中，您可以使用 Elastic Beanstalk 在环境实例上提供的 `get-config` 实用程序。有关详细信息，请参阅 the section called “平台脚本工具”。

配置文件

您可以将配置文件添加到应用程序源代码的 `.ebextensions` 目录中，以配置 Elastic Beanstalk 环境的各个方面。此外，配置文件还允许您自定义环境实例上的软件和其他文件，并在实例上运行初始化命令。有关更多信息，请参阅the section called “Linux 服务器”。

您还可以使用配置文件设置配置选项。许多选项控制平台行为，其中的某些选项是平台特定选项。

对于基于 Amazon Linux 2 和 Amazon Linux 2023 的平台，我们建议在实例预置期间使用 Buildfile、Procfile 和平台挂钩在环境实例上配置和运行自定义代码。本页上前面的部分中描述了这些机制。您仍可以在 `.ebextensions` 配置文件中使用命令和容器命令，但这并不简单。例如，从语法角度而言，在 YAML 文件中编写命令脚本可能面临挑战。对于需要引用 AWS CloudFormation 资源的任何脚本，您仍需要使用 `.ebextensions` 配置文件。

反向代理配置

所有 Amazon Linux 2 和 Amazon Linux 2023 平台版本都使用 nginx 作为其默认的反向代理服务器。Tomcat、Node.js、PHP 和 Python 平台也支持将 Apache HTTPD 作为替代方案。要在这些平台上选择 Apache，请将 `aws:elasticbeanstalk:environment:proxy` 命名空间中的 `ProxyServer` 选项设置为 `apache`。所有平台都以一致的方式启用代理服务器配置，如本节所述。

> **Note**
>
> 在 Amazon Linux AMI 平台版本（Amazon Linux 2 之前的版本），您需要以不同的方式配置代理服务器。您可以在本指南中的相应平台主题下找到这些旧的详细信息。

Elastic Beanstalk 在环境实例上将代理服务器配置为向环境根 URL 的主要 Web 应用程序转发 Web 流量；例如，`http://my-env.elasticbeanstalk.com`。

默认情况下，Elastic Beanstalk 将代理配置为通过端口 5000 向主要 Web 应用程序转发 80 端口上的传入请求。通过使用配置文件中的 aws:elasticbeanstalk:application:environment 命名空间来设置 PORT 环境属性，您可以配置此端口号，如以下示例所示。

```
option_settings:
  - namespace:  aws:elasticbeanstalk:application:environment
    option_name:  PORT
    value:  <main_port_number>
```

有关设置应用程序环境变量的更多信息，请参阅the section called “选项设置”。

您的应用程序应侦听代理中为其配置的端口。如果使用 PORT 环境属性更改默认端口，代码可以通过读取 PORT 环境变量的值来访问该端口。例如，在 Go 中调用 `os.Getenv("PORT")`，或者在 Java 中调用 `System.getenv("PORT")`。如果您将代理配置为向多个应用程序进程发送流量，则可以配置多个环境属性，并在代理配置和应用程序代码中使用它们的值。另一种选择是将端口值作为 `Procfile` 中的命令参数传递给进程。有关详细信息，请展开此页上的 Buildfile 和 Procfile 部分。

配置 nginx

Elastic Beanstalk 使用 nginx 作为默认反向代理，将应用程序映射到 Elastic Load Balancing 负载均衡器。Elastic Beanstalk 提供一个默认 nginx 配置，您可以扩展该配置，或者将其完全替换为您自己的配置。

> **Note**
>
> 添加或编辑 nginx `.conf` 配置文件时，请务必将其编码为 UTF-8。

要扩展 Elastic Beanstalk 的默认 nginx 配置，请将 `.conf` 配置文件添加到您的应用程序源包的 `.platform/nginx/conf.d/` 文件夹中。Elastic Beanstalk nginx 配置自动在此文件夹中包括 `.conf` 文件。

```
~/workspace/my-app/
|-- .platform
|   `-- nginx
|       `-- conf.d
|           `-- myconf.conf
`-- other source files
```

要完全覆盖 Elastic Beanstalk 默认 nginx 配置，请在您的源包的 `.platform/nginx/nginx.conf` 处包含一个配置：

```
~/workspace/my-app/
|-- .platform
|   `-- nginx
|       `-- nginx.conf
`-- other source files
```

如果要覆盖 Elastic Beanstalk nginx 配置，请将以下行添加到 `nginx.conf`，以便加入适用于 增强型运行状况报告和监控、自动应用程序映射和静态文件的 Elastic Beanstalk 配置。

```
 include conf.d/elasticbeanstalk/*.conf;
```

配置 Apache HTTPD

Tomcat、Node.js、PHP 和 Python 平台允许您选择 Apache HTTPD 代理服务器作为 nginx 的替代方案。这不是默认值。以下示例将 Elastic Beanstalk 配置为使用 Apache HTTPD。

Example .ebextensions/httpd-proxy.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
```

您可以使用其他配置文件扩展 Elastic Beanstalk 默认 Apache 配置。也可以完全覆盖 Elastic Beanstalk 默认 Apache 配置。

要扩展 Elastic Beanstalk 默认 Apache 配置，请将 `.conf` 配置文件添加到应用程序源包中名为 `.platform/httpd/conf.d` 的文件夹中。Elastic Beanstalk Apache 配置自动在此文件夹中包括 `.conf` 文件。

```
~/workspace/my-app/
|-- .ebextensions
|   -- httpd-proxy.config
|-- .platform
|   -- httpd
|      -- conf.d
|         -- port5000.conf
|         -- ssl.conf
-- index.jsp
```

例如，以下 Apache 2.4 配置将在端口 5000 上添加一个监听器。

Example .platform/httpd/conf.d/port5000.conf

```
listen 5000
<VirtualHost *:5000>
  <Proxy *>
    Require all granted
  </Proxy>
  ProxyPass / http://localhost:8080/ retry=0
```

```
  ProxyPassReverse / http://localhost:8080/
  ProxyPreserveHost on

  ErrorLog /var/log/httpd/elasticbeanstalk-error_log
</VirtualHost>
```

要完全覆盖 Elastic Beanstalk 默认 Apache 配置，请在源包的 `.platform/httpd/conf/httpd.conf` 处包括一个配置。

```
~/workspace/my-app/
|-- .ebextensions
|   -- httpd-proxy.config
|-- .platform
|   `-- httpd
|       `-- conf
|           `-- httpd.conf
`-- index.jsp
```

> **Note**
>
> 如果要覆盖 Elastic Beanstalk Apache 配置，请将以下行添加到 `httpd.conf`，以便加入适用于 增强型运行状况报告和监控、自动应用程序映射和静态文件的 Elastic Beanstalk 配置。
>
> ```
> IncludeOptional conf.d/elasticbeanstalk/*.conf
> ```

> **Note**
>
> 如果您要将 Elastic Beanstalk 应用程序迁移到 Amazon Linux 2 或 Amazon Linux 2023 平台，请务必阅读 the section called “迁移到 AL2023/AL2” 中的信息。

主题

- 带扩展功能的应用程序示例
- 实例部署工作流
- 在 Amazon Linux 2 及更高版本上运行的 ECS 的实例部署工作流
- 平台脚本工具

带扩展功能的应用程序示例

以下示例演示一个应用程序源包，其中包含 Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台支持的多个可扩展性功能：`Procfile`、`.ebextensions` 配置文件、自定义挂钩和代理配置文件。

```
~/my-app/
|-- web.jar
|-- Procfile
|-- readme.md
|-- .ebextensions/
|   |-- options.config        # Option settings
|   `-- cloudwatch.config     # Other .ebextensions sections, for example files and
 container commands
`-- .platform/
    |-- nginx/                # Proxy configuration
    |   |-- nginx.conf
    |   `-- conf.d/
    |       `-- custom.conf
    |-- hooks/                # Application deployment hooks
    |   |-- prebuild/
    |   |   |-- 01_set_secrets.sh
    |   |   `-- 12_update_permissions.sh
    |   |-- predeploy/
    |   |   `-- 01_some_service_stop.sh
    |   `-- postdeploy/
    |       |-- 01_set_tmp_file_permissions.sh
    |       |-- 50_run_something_after_app_deployment.sh
    |       `-- 99_some_service_start.sh
    `-- confighooks/          # Configuration deployment hooks
        |-- prebuild/
        |   `-- 01_set_secrets.sh
        |-- predeploy/
        |   `-- 01_some_service_stop.sh
        `-- postdeploy/
            |-- 01_run_something_after_config_deployment.sh
            `-- 99_some_service_start.sh
```

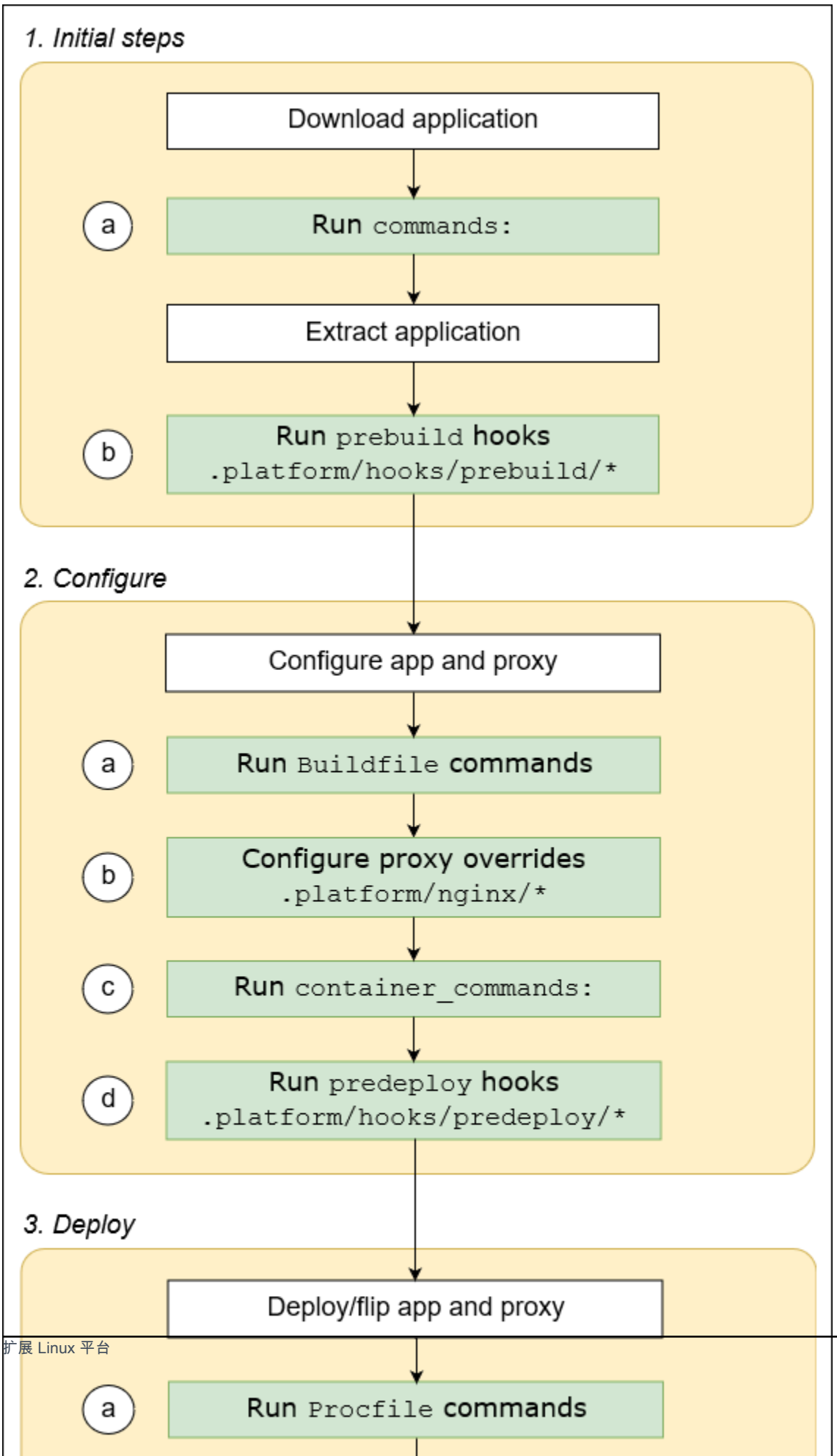
> **Note**
>
> Amazon Linux AMI 平台版本（Amazon Linux 2 以前的版本）不支持其中一些扩展。

实例部署工作流

> **Note**
>
> 本节信息不适用于在 Amazon Linux 2 和 Amazon Linux 2023 上运行的 ECS 平台分支。有关更多信息，请参阅下一部分在 Amazon Linux 2 及更高版本上运行的 ECS 的实例部署工作流。

有多种扩展环境平台的方法，对于了解 Elastic Beanstalk 在预配置实例或向实例运行部署时会发生什么情况非常有用。下图显示了整个部署工作流程。它描述了部署中的不同阶段以及 Elastic Beanstalk 在每个阶段中采取的步骤。

> **注意**
>
> - 该图不代表 Elastic Beanstalk 在部署期间对环境实例采取的完整步骤集。我们提供此图作为说明，为您提供执行自定义项的顺序和上下文。
> - 为简单起见，图中仅提及 `.platform/hooks/*` 挂钩子目录（用于应用程序部署），而不提及 `.platform/confighooks/*` 挂钩子目录（用于配置部署）。后面子目录中的挂钩运行的步骤与图中显示的相应子目录中的挂钩运行的步骤完全相同。


1. Initial steps
Download application
a
Run commands:
Extract application
b
Run prebuild hooks
.platform/hooks/prebuild/*
2. Configure
Configure app and proxy
a
Run Buildfile commands
b
Configure proxy overrides
.platform/nginx/*
c
Run container_commands:
d
Run predeploy hooks
.platform/hooks/predeploy/*
3. Deploy
Deploy/flip app and proxy
a
Run Procfile commands

以下列表详细介绍了部署阶段和步骤。

1. 初始步骤

   Elastic Beanstalk 下载并提取您的应用程序。在上述每个步骤之后，Elastic Beanstalk 会运行一个可扩展性步骤。

   a. 运行任何配置文件的 commands: 部分中的命令。

   b. 运行在源包的 `.platform/hooks/prebuild` 目录中找到的任何可执行文件（`.platform/confighooks/prebuild` 用于配置部署）。

2. 配置

   Elastic Beanstalk 配置您的应用程序和代理服务器。

   a. 运行源包 `Buildfile` 中的命令。

   b. 如果源包 `.platform/nginx` 目录中包含任何自定义代理配置文件，请将其复制到其运行时位置。

   c. 运行任何配置文件的 container_commands: 部分中的命令。

   d. 运行在源包的 `.platform/hooks/predeploy` 目录中找到的任何可执行文件（`.platform/confighooks/predeploy` 用于配置部署）。

3. 部署

   Elastic Beanstalk 部署并运行您的应用程序和代理服务器。

   a. 运行源包 `Procfile` 文件中的命令。

   b. 使用您的自定义代理配置文件（如果有）运行或重新运行代理服务器。

   c. 运行在源包的 `.platform/hooks/postdeploy` 目录中找到的任何可执行文件（`.platform/confighooks/postdeploy` 用于配置部署）。

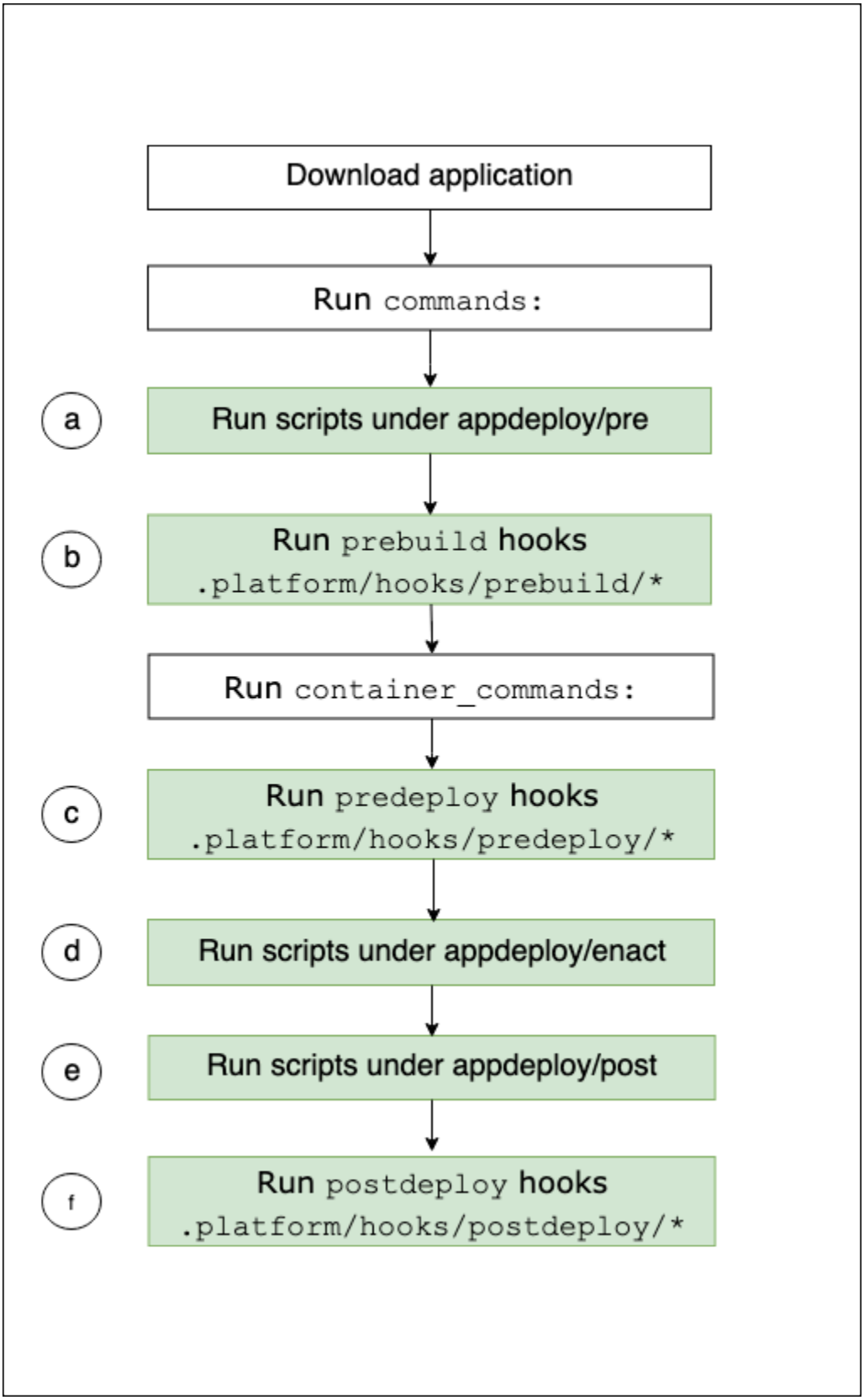
在 Amazon Linux 2 及更高版本上运行的 ECS 的实例部署工作流

上一节介绍了应用程序部署工作流各个阶段支持的可扩展性功能。Docker 平台分支在 Amazon Linux 2 及更高版本上运行的 ECS有一些不同之处。本节介绍这些概念如何应用于此特定平台分支。

有多种扩展环境平台的方法，对于了解 Elastic Beanstalk 在预配置实例或向实例运行部署时会发生什么情况非常有用。下图显示了基于在 Amazon Linux 2 上运行的 ECS 和在 Amazon Linux 2023 上运行的 ECS 平台分支的环境的整个部署工作流。它描述了部署中的不同阶段以及 Elastic Beanstalk 在每个阶段中采取的步骤。

与上一节中描述的工作流不同，部署配置阶段不支持以下可扩展性功能：Buildfile命令、Procfile命令、反向代理配置。

> **注意**
>
> - 该图不代表 Elastic Beanstalk 在部署期间对环境实例采取的完整步骤集。我们提供此图作为说明，为您提供执行自定义项的顺序和上下文。
> - 为简单起见，图中仅提及 `.platform/hooks/*` 挂钩子目录（用于应用程序部署），而不提及 `.platform/confighooks/*` 挂钩子目录（用于配置部署）。后面子目录中的挂钩运行的步骤与图中显示的相应子目录中的挂钩运行的步骤完全相同。



以下列表详细介绍了部署工作流步骤。

a. 运行 EBhooksDir 下 appdeploy/pre 目录中找到的任何可执行文件。
b. 运行在源包的 .platform/hooks/prebuild 目录中找到的任何可执行文件（.platform/confighooks/prebuild 用于配置部署）。
c. 运行在源包的 .platform/hooks/predeploy 目录中找到的任何可执行文件（.platform/confighooks/predeploy 用于配置部署）。
d. 运行 EBhooksDir 下 appdeploy/enact 目录中找到的任何可执行文件。
e. 运行 EBhooksDir 下 appdeploy/post 目录中找到的任何可执行文件。
f. 运行在源包的 .platform/hooks/postdeploy 目录中找到的任何可执行文件（.platform/confighooks/postdeploy 用于配置部署）。

对 EBhooksDir 的引用表示平台挂钩目录的路径。要检索目录路径名，请使用环境实例命令行上的 get-config 脚本工具，如下所示：

```
$ /opt/elasticbeanstalk/bin/get-config platformconfig -k EBhooksDir
```

平台脚本工具

本主题介绍 AWS Elastic Beanstalk 为使用 Amazon Linux 平台的环境提供的工具。这些工具位于 Elastic Beanstalk 环境的 Amazon EC2 实例上。

get-config

使用 get-config 工具检索环境变量值以及其他平台和实例信息。可在 /opt/elasticbeanstalk/bin/get-config 中获得此工具。

get-config 命令

每个 get-config 工具命令都返回特定类型的信息。使用以下语法运行任何工具的命令。

```
$ /opt/elasticbeanstalk/bin/get-config command [ options ]
```

以下示例运行 environment 命令。

```
$ /opt/elasticbeanstalk/bin/get-config environment -k PORT
```

根据您选择的命令和选项，工具返回具有键值对或单个值的对象（JSON 或 YAML）。

您可以通过使用 SSH 连接到 Elastic Beanstalk 环境中的 EC2 实例来测试 get-config。

> **Note**
>
> 运行 get-config 以进行测试时，某些命令可能需要 root 用户权限才能访问基础信息。如果您收到访问权限错误，请在 sudo 下再次运行命令。
> 在部署到环境的脚本中使用该工具时，无需添加 sudo。Elastic Beanstalk 以 root 用户身份运行所有脚本。

以下各节介绍这些工具的命令。

optionsettings – 配置选项

get-config optionsettings 命令返回一个对象，其中列出在环境上设置并由平台在环境实例上使用的配置选项。它们按命名空间排列。

```
$ /opt/elasticbeanstalk/bin/get-config optionsettings
{"aws:elasticbeanstalk:application:environment":
{"JDBC_CONNECTION_STRING":""},"aws:elasticbeanstalk:container:tomcat:jvmoptions":{"JVM
 Options":"","Xms":"256m","Xmx":"256m"},"aws:elasticbeanstalk:environment:proxy":
{"ProxyServer":"nginx","StaticFiles":
[""]},"aws:elasticbeanstalk:healthreporting:system":
{"SystemType":"enhanced"},"aws:elasticbeanstalk:hostmanager":
{"LogPublicationControl":"false"}}
```

要返回特定配置选项值，请使用 --namespace (-n) 选项来指定命名空间，并使用 --option-name (-o) 选项来指定选项名称。

```
$ /opt/elasticbeanstalk/bin/get-config optionsettings -
n aws:elasticbeanstalk:container:php:phpini -o memory_limit
256M
```

environment – 环境属性

get-config environment 命令返回一个包含环境属性列表的对象。这些属性包括用户配置的属性和由 Elastic Beanstalk 提供的属性。

```
$ /opt/elasticbeanstalk/bin/get-config environment
{"JDBC_CONNECTION_STRING":"","RDS_PORT":"3306","RDS_HOSTNAME":"anj9aw1b0tbj6b.cijbpanmxz5u.us-
west-2.rds.amazonaws.com","RDS_USERNAME":"testusername","RDS_DB_NAME":"ebdb","RDS_PASSWORD":"te
```

例如，Elastic Beanstalk 提供用于连接到集成 Amazon RDS 数据库实例的环境属性（例如，RDS_HOSTNAME）。这些 RDS 连接属性在 `get-config environment` 的输出中显示。但是，它们不会在 `get-config optionsettings` 的输出中显示。这是因为没有在配置选项中设置这些属性。

要返回特定环境属性，请使用 `--key (-k)` 选项来指定属性键。

```
$ /opt/elasticbeanstalk/bin/get-config environment -k TESTPROPERTY
testvalue
```

container – 实例上的配置值

`get-config container` 命令返回一个对象，其中列出环境实例的平台和环境配置值。

以下示例显示此命令在 Amazon Linux 2 Tomcat 环境上的输出。

```
$ /opt/elasticbeanstalk/bin/get-config container
{"common_log_list":["/var/log/eb-engine.log","/var/log/eb-
hooks.log"],"default_log_list":["/var/log/nginx/access.log","/var/log/nginx/
error.log"],"environment_name":"myenv-1da84946","instance_port":"80","log_group_name_prefix":"/
aws/elasticbeanstalk","proxy_server":"nginx","static_files":
[""],"xray_enabled":"false"}
```

要返回特定键的值，请使用 `--key (-k)` 选项指定键。

```
$ /opt/elasticbeanstalk/bin/get-config container -k environment_name
myenv-1da84946
```

addons – 附加项配置值

`get-config addons` 命令返回一个包含环境附加项配置信息的对象。使用它检索与环境关联的 Amazon RDS 数据库的配置。

```
$ /opt/elasticbeanstalk/bin/get-config addons
{"rds":{"Description":"RDS Environment variables","env":
{"RDS_DB_NAME":"ebdb","RDS_HOSTNAME":"ea13k2wimu1dh8i.c18mnpu5rwvg.us-
east-2.rds.amazonaws.com","RDS_PASSWORD":"password","RDS_PORT":"3306","RDS_USERNAME":"user"}}}
```

您可以通过两种方式限制结果。要检索特定附加项的值，请使用 `--add-on (-a)` 选项指定附加项名称。

```
$ /opt/elasticbeanstalk/bin/get-config addons -a rds
{"Description":"RDS Environment variables","env":
{"RDS_DB_NAME":"ebdb","RDS_HOSTNAME":"ea13k2wimu1dh8i.c18mnpu5rwvg.us-
east-2.rds.amazonaws.com","RDS_PASSWORD":"password","RDS_PORT":"3306","RDS_USERNAME":"user"}}
```

要返回附加项中特定密钥的值，请添加 `--key (-k)` 选项以指定密钥。

```
$ /opt/elasticbeanstalk/bin/get-config addons -a rds -k RDS_DB_NAME
ebdb
```

platformconfig – 恒定的配置值

`get-config platformconfig` 命令返回一个对象，其中包含对平台版本恒定不变的平台配置信息。在运行相同平台版本的所有环境中，输出是相同的。命令的输出对象有两个嵌入式对象：

- `GeneralConfig` – 包含在所有 Amazon Linux 2 和 Amazon Linux 2023 平台分支的最新版本中恒定不变的信息。
- `PlatformSpecificConfig` – 包含对于平台版本而言恒定不变且特定于该版本的信息。

以下示例显示此命令在使用运行 Corretto 11 平台分支的 Tomcat 8.5 的环境上的输出。

```
$ /opt/elasticbeanstalk/bin/get-config platformconfig
{"GeneralConfig":{"AppUser":"webapp","AppDeployDir":"/var/app/
current/","AppStagingDir":"/var/app/
staging/","ProxyServer":"nginx","DefaultInstancePort":"80"},"PlatformSpecificConfig":
{"ApplicationPort":"8080","JavaVersion":"11","TomcatVersion":"8.5"}}
```

要返回特定键的值，请使用 `--key (-k)` 选项指定键。这些键在两个嵌入式对象间是唯一的。您不需要指定包含密钥的对象。

```
$ /opt/elasticbeanstalk/bin/get-config platformconfig -k AppStagingDir
/var/app/staging/
```

get-config 输出选项

使用 `--output` 选项指定输出对象格式。有效值为 `JSON`（默认值）和 `YAML`。这是一个全局选项。您必须在命令名称之前指定该选项。

以下示例以 YAML 格式返回配置选项值。

```
$ /opt/elasticbeanstalk/bin/get-config --output YAML optionsettings
aws:elasticbeanstalk:application:environment:
  JDBC_CONNECTION_STRING: ""
aws:elasticbeanstalk:container:tomcat:jvmoptions:
  JVM Options: ""
  Xms: 256m
  Xmx: 256m
aws:elasticbeanstalk:environment:proxy:
  ProxyServer: nginx
  StaticFiles:
        - ""
aws:elasticbeanstalk:healthreporting:system:
  SystemType: enhanced
aws:elasticbeanstalk:hostmanager:
  LogPublicationControl: "false"
```

pkg-repo

> **Note**
>
> pkg-repo 工具不适用于基于 Amazon Linux 2023 平台的环境。但是，您可以手动将软件包和操作系统更新应用于 AL2023 实例。有关更多信息，请参阅 Amazon Linux 2023 用户指南中的管理软件包和操作系统更新

在某些紧急情况下，您可能需要使用尚未与所需的 Elastic Beanstalk 平台版本一起发布的 Amazon Linux 2 安全补丁来更新您的 Amazon EC2 实例。默认情况下，您无法对 Elastic Beanstalk 环境执行手动更新。这是因为平台版本被锁定到特定版本的 Amazon Linux 2 存储库。此锁定可确保实例运行受支持且一致的软件版本。对于紧急情况，如果您需要在新的 Elastic Beanstalk 平台版本中发布之前在环境中安装该程序包，则 pkg-repo 工具允许在 Amazon Linux 2 上手动更新 yum 程序包作为解决方法。

Amazon Linux 2 平台上的 pkg-repo 工具提供解锁 yum 程序包存储库的功能。然后，您可以对安全补丁手动执行 yum update。相反，您可以通过使用该工具锁定 yum 程序包存储库来跟踪更新，以防止进一步更新。Elastic Beanstalk 环境中所有 EC2 实例的 /opt/elasticbeanstalk/bin/pkg-repo 目录中均提供 pkg-repo 工具。

使用 pkg-repo 工具的更改只能在使用该工具的 EC2 实例上进行。这些更改不会影响其他实例，也不会阻止将来对环境进行更新。本主题后面提供的示例说明如何通过从脚本和配置文件调用 pkg-repo 命令来跨所有实例应用更改。

> **Warning**
>
> 我们不建议大多数用户使用此工具。应用于已解锁平台版本的任何手动更改都将被视为带外更改。此选项仅适用于在紧急情况下可接受以下风险的用户：
>
> - 无法保证程序包版本在您的环境中的所有实例之间保持一致。
> - 使用 pkg-repo 工具修改的环境无法保证正常运行，因为这些环境尚未在 Elastic Beanstalk 支持的平台上进行测试和验证。
>
> 我们强烈建议应用包括测试和撤销计划的最佳实践。为帮助促进最佳实践，您可以使用 Elastic Beanstalk 控制台和 EB CLI 克隆环境并交换环境 URL。有关使用这些操作的更多信息，请参阅本指南管理环境一章中的蓝绿部署。

如果您计划手动编辑 yum 存储库配置文件，请首先运行 pkg-repo 工具。采用手动编辑的 yum 存储库配置文件的 pkg-repo 工具在 Amazon Linux 2 环境中可能无法按预期工作。这是因为该工具可能无法识别配置更改。

有关 Amazon Linux 软件包存储库的更多信息，请参阅 Amazon EC2 用户指南中的软件包存储库主题。

pkg-repo 命令

使用以下语法运行 pkg-repo 工具命令。

```
$ /opt/elasticbeanstalk/bin/pkg-repo command [options]
```

pkg-repo 命令如下：

- lock – 将 yum 程序包存储库锁定到特定版本
- unlock – 从特定版本解锁 yum 程序包存储库
- status – 列出所有 yum 程序包存储库及其当前锁定状态
- help – 显示通用帮助或一条命令的帮助

这些选项适用于以下命令：

- lock、unlock 和 status –选项：-h、--help 或无（默认值）。

- `help` – 选项：`lock`、`unlock`、`status` 或无（默认值）。

以下示例运行 unlock 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo unlock
Amazon Linux 2 core package repo successfully unlocked
Amazon Linux 2 extras package repo successfully unlocked
```

以下示例运行 lock 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo lock
Amazon Linux 2 core package repo successfully locked
Amazon Linux 2 extras package repo successfully locked
```

以下示例运行 status 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo status
Amazon Linux 2 core package repo is currently UNLOCKED
Amazon Linux 2 extras package repo is currently UNLOCKED
```

以下示例运行 lock 命令的 help 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo help lock
```

以下示例运行 `pkg-repo` 工具的 help 命令。

```
$ sudo /opt/elasticbeanstalk/bin/pkg-repo help
```

您可以通过使用 SSH 连接到 Elastic Beanstalk 环境中的实例来测试 `pkg-repo`。一个 SSH 选项是 EB CLI eb ssh 命令。

> **Note**
>
> `pkg-repo` 工具需要根用户权限才能运行。如果您收到访问权限错误，请在 `sudo` 下再次运行命令。
> 在部署到环境的脚本或配置文件中使用该工具时，无需添加 `sudo`。Elastic Beanstalk 以 root 用户身份运行所有脚本。

pkg-repo 示例

上一节提供用于在 Elastic Beanstalk 环境的单个 EC2 实例上进行测试的命令行示例。这种方法对测试很有帮助。但是，它一次只更新一个实例，因此将更改应用到环境中的所有实例是不切实际的。

更实用的方法是使用平台挂钩脚本或 `.ebextensions` 配置文件以一致的方式跨所有实例应用更改。

以下示例从 `.ebextensions` 文件夹中的配置文件调用 `pkg-repo`。当您部署应用程序源代码捆绑包时，Elastic Beanstalk 将运行 `update_package.config` 文件中的命令。

```
.ebextensions
### update_package.config
```

要接收最新版本的 docker 程序包，此配置将在 yum update 命令中指定 docker 程序包。

```
### update_package.config ###

commands:
  update_package:
    command: |
      /opt/elasticbeanstalk/bin/pkg-repo unlock
      yum update docker -y
      /opt/elasticbeanstalk/bin/pkg-repo lock
      yum clean all -y
      rm -rf /var/cache/yum
```

此配置没有在 yum update 命令中指定任何程序包，因此将应用所有可用的更新。

```
### update_package.config ###

commands:
  update_package:
    command: |
      /opt/elasticbeanstalk/bin/pkg-repo unlock
      yum update -y
      /opt/elasticbeanstalk/bin/pkg-repo lock
      yum clean all -y
      rm -rf /var/cache/yum
```

以下示例从 bash 脚本调用 `pkg-repo` 作为平台挂钩。Elastic Beanstalk 运行位于 `prebuild` 子目录中的 `update_package.sh` 脚本文件。

```
.platform
### hooks
    ### prebuild
        ### update_package.sh
```

要接收最新版本的 docker 程序包，此脚本将在 yum update 命令中指定 docker 程序包。如果省略软件包名称，将应用所有可用的更新。前面的配置文件示例展示了这种情况。

```
### update_package.sh ###

#!/bin/bash

/opt/elasticbeanstalk/bin/pkg-repo unlock
yum update docker -y
/opt/elasticbeanstalk/bin/pkg-repo lock
yum clean all -y
rm -rf /var/cache/yum
```

download-source-bundle （仅限亚马逊 Linux AMI）

在 Amazon Linux AMI 平台分支（Amazon Linux 2 以前的版本）上，Elastic Beanstalk 提供了额外的工具，即 `download-source-bundle`。部署平台时，使用此工具下载应用程序源代码。可在 `/opt/elasticbeanstalk/bin/download-source-bundle` 中获得此工具。

示例脚本 `00-unzip.sh` 位于环境实例上的 `appdeploy/pre` 文件夹中。它演示在部署过程中如何使用 `download-source-bundle` 将应用程序源代码下载到 `/opt/elasticbeanstalk/deploy/appsource` 文件夹。

从 Docker 容器部署 Elastic Beanstalk 应用程序

本章介绍如何使用 Elastic Beanstalk 从 Docker 容器中部署 Web 应用程序。Docker 容器具有独立性且包含您的 Web 应用程序运行所需的所有配置信息和软件。使用 Docker 容器，您可以定义自己的运行时环境。您还可以选择自己的编程语言和应用程序依赖项，例如包管理器或工具，而其他 Elastic Beanstalk 平台通常不支持这些内容。

按照中的QuickStart 适用于 Docker步骤创建 Docker “Hello World” 应用程序，然后使用 EB CLI 将其部署到 Elastic Beanstalk 环境。

主题

Docker 平台分支

Elastic Beanstalk Docker 平台支持以下平台分支：

运行 Amazon Linux 2 的 Docker 和运行 AL2023 的 Docker

Elastic Beanstalk 将 Docker 容器和源代码部署到 EC2 实例并对其进行管理。这些平台分支提供多容器支持。您可以利用 Docker Compose 工具来简化应用程序配置、测试和部署。有关此平台分支的更多信息，请参阅 the section called “Docker 平台分支”。

在 Amazon Linux 2 上运行的 ECS和在 AL2023 上运行的 ECS

我们为需要一个从已停用的平台分支 Amazon Linux AMI 上运行的多容器 Docker 迁移到 AL2023/AL2 的路径的客户提供此分支。最新的平台分支支持已停用平台分支的所有功能。源代码无需更改。有关更多信息，请参阅 将在 Amazon Linux 上运行的多容器 Docker 迁移到 Amazon Linux 2023 上的 ECS。如果您没有在基于 ECS 的平台分支上运行 Elastic Beanstalk 环境，则建议您使用该平台分支，即在 64 位 AL2023 上运行的 Docker。这提供了一种更简单的方法，需要的资源也更少。

在 Amazon Linux AMI（AL1）上运行的已停用平台分支

2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。展开以下各节，详细了解每个已停用的平台分支及其向 Amazon Linux 2 或 Amazon Linux 2023（推荐）上运行的最新平台分支的迁移路径。

Docker（Amazon Linux AMI）

此平台分支可以部署 Docker 映像，如 Dockerfile 或 `Dockerrun.aws.json` v1 定义所述。此平台分支对于每个实例只运行一个容器。它的后续平台分支在 64 位 AL2023 上运行的 Docker 和在 64 位 Amazon Linux 2 上运行的 Docker 对于每个实例支持多个 Docker 容器。

我们建议您创建环境使用较新且受支持的平台分支在 64 位 AL2023 上运行的 Docker。然后，您可以将应用程序迁移到新创建的环境。有关创建这些环境的更多信息，请参阅 the section called “Docker 平台分支”。有关迁移的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

多容器 Docker（Amazon Linux AMI）

此平台分支使用 Amazon ECS 将多个 Docker 容器协调部署到 Elastic Beanstalk 环境中的 Amazon ECS 集群。如果您目前正在使用这个已停用的平台分支，我们建议您迁移至最新的平台分支：在 Amazon Linux 2023 上运行的 ECS。最新的平台分支支持此已停产平台分支的所有功能。源代码无需更改。有关更多信息，请参阅 将在 Amazon Linux 上运行的多容器 Docker 迁移到 Amazon Linux 2023 上的 ECS。

预配置 Docker 容器

除了前面提到的 Docker 平台外，还有在亚马逊 Linux AMI 操作系统 (AL1) 上运行的预配置 Docker GlassFish 平台分支。

该平台分支已被平台分支在 64 位 AL2023 上运行的 Docker 和在 64 位 Amazon Linux 2 上运行的 Docker 所取代。有关更多信息，请参阅将 GlassFish 应用程序部署到 Docker 平台。

使用 Docker 平台分支

AWS Elastic Beanstalk 可以通过构建中描述的映像`Dockerfile`或拉取远程 Docker 镜像来启动 Docker 环境。如果您要部署远程 Docker 映像，则无需包含 `Dockerfile`。相反，如果您也使用 Docker Compose，则使用 `docker-compose.yml` 文件，该文件会指定要使用的映像和其他配置选项。如果您没有将 Docker Compose 与 Docker 环境结合使用，请改为使用 `Dockerrun.aws.json` 文件。

主题

- QuickStart: 将 Docker 应用程序部署到 Elastic Beanstalk
- Docker 配置
- 配置 Docker 环境

QuickStart: 将 Docker 应用程序部署到 Elastic Beanstalk

本 QuickStart 教程将引导您完成创建 Docker 应用程序并将其部署到 AWS Elastic Beanstalk 环境的过程。

> **Note**
>
> 本 QuickStart 教程仅用于演示目的。请勿将本教程中创建的应用程序用于生产流量。

Sections

你的 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

如果您已经有一个 AWS 帐户，则可以继续前进先决条件 。

创建一个 AWS 账户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

先决条件

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

EB CLI

本教程使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

Docker

要学习本教程，你需要在本地安装 Docker。有关更多信息，请参阅 Docker 文档网站上的获取 Docker。

运行以下命令，验证 Docker 守护程序是否已启动并正在运行。

```
~$ docker info
```

步骤 1：创建 Docker 应用程序和容器

在本示例中，我们创建了示例 Flask 应用程序的 Docker 镜像，中也引用了该镜像。将 Flask 应用程序部署到 Elastic Beanstalk

该应用程序由两个文件组成：

- `app.py`— 包含将在容器中执行的代码的 Python 文件。
- `Dockerfile`— 用于构建镜像的 Dockerfile。

将两个文件都放在目录的根目录下。

```
~/eb-docker-flask/
|-- Dockerfile
|-- app.py
```

将以下内容添加到您的Dockerfile。

Example **~/eb-docker-flask/Dockerfile**

```
FROM python:3.12
COPY . /app
WORKDIR /app
RUN pip install Flask==3.0.2
EXPOSE 5000
CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

将以下内容添加到您的app.py文件中。

Example **~/eb-docker-flask/app.py**

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello Elastic Beanstalk! This is a Docker application'
```

构建 Docker 容器，用标记镜像。eb-docker-flask

```
~/eb-docker-flask$ docker build -t eb-docker-flask
```

步骤 2：在本地运行应用程序

使用 docker build 命令在本地构建容器镜像，并用标记镜像。eb-docker-flask命令末尾的句点 (.) 指定该路径是本地目录。

```
~/eb-docker-flask$ docker run -dp 127.0.0.1:5000:5000 eb-docker-flask .
```

使用 docker run 命令运行你的容器。该命令将打印正在运行的容器的 ID。该-d选项在后台模式下运行 docker。该-p选项在端口 5000 上公开您的应用程序。默认情况下，Elastic Beanstalk 在 Docker 平台上为端口 5000 提供流量。

```
~/eb-docker-flask$ docker run -dp 127.0.0.1:5000:5000 eb-docker-flask container-id
```

http://127.0.0.1:5000/ 在浏览器中导航至。你应该会看到文字 “你好 Elastic Beanstalk！这是一个 Docker 应用程序”。

运行 d ocker kill 命令终止容器。

```
~/eb-docker-flask$ docker kill container-id
```

步骤 3：使用 EB CLI 部署您的 Docker 应用程序

运行以下命令为此应用程序创建 Elastic Beanstalk 环境。

创建环境并部署 Docker 应用程序

1. 使用 eb init 命令，初始化 EB CLI 存储库。

   ```
   ~/eb-docker-flask$ eb init -p docker docker-tutorial us-east-2
   Application docker-tutorial has been created.
   ```

   此命令将创建一个名为的应用程序，docker-tutorial并将您的本地存储库配置为使用最新 Docker 平台版本创建环境。
2. （可选）再次运行 eb init 以配置默认密钥对，以便使用 SSH 连接到运行您的应用程序的 EC2 实例。

   ```
   ~/eb-docker-flask$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   如果您已有密钥对，请选择一个，或按提示创建一个。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。
3. 创建环境并使用 eb create 将应用程序部署到此环境中。Elastic Beanstalk 会自动为您的应用程序生成一个 zip 文件并在端口 5000 上启动该文件。

   ```
   ~/eb-docker-flask$ eb create docker-tutorial
   ```

Elastic Beanstalk 创建您的环境大约需要五分钟。

第 4 步：在 Elastic Beanstalk 上运行你的应用程序

创建环境的过程完成后，使用打开您的网站eb open。

```
~/eb-docker-flask$ eb open
```

恭喜您！你已经使用 Elastic Beanstalk 部署了 Docker 应用程序！这将使用为应用程序创建的域名打开一个浏览器窗口。

第 5 步：清理

完成应用程序的使用后，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源。

要使用 EB CLI 终止您的 Elastic Beanstalk 环境，请运行以下命令。

```
~/eb-docker-flask$ eb terminate
```

AWS 您的应用程序的资源

您刚刚创建了一个单实例应用程序。它可用作带有单个 EC2 实例的简单示例应用程序，因此不需要负载平衡或 auto Scaling。对于单实例应用程序，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

  各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。

- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 管理所有这些资源。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。您还可以使用 Elastic Beanstalk 控制台探索您的新环境。有关详细步骤，请参阅本指南入门一章中的探索您的环境。

部署一两个示例应用程序并准备好开始在本地开发和运行 Docker 应用程序后，请参阅

使用 Elastic Beanstalk 控制台进行部署

您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。

Docker 配置

本节介绍如何为部署到 Elastic Beanstalk 准备 Docker 映像和容器。

带有 Docker Compose 的 Docker 环境

本节介绍如何为部署到 Elastic Beanstalk 准备 Docker 映像和容器。如果您还使用 Docker Compose 工具，您在 Docker 环境中部署到 Elastic Beanstalk 的任何 Web 应用程序都必须包含一个 `docker-compose.yml` 文件。您可以通过执行以下操作之一将 Web 应用程序作为容器化服务部署到 Elastic Beanstalk：

- 创建一个用于将 Docker 映像从托管存储库部署到 Elastic Beanstalk 的 `docker-compose.yml` 文件。如果所有部署都来自公有存储库中的映像，则不需要其他文件。（如果您的部署必须从专用存储库获取映像，则需要包含其他配置文件以进行身份验证。有关更多信息，请参阅使用私有存储库中的映像。）有关 `docker-compose.yml` 文件的更多信息，请参阅 Docker 网站上的 Compose 文件参考。
- 创建 `Dockerfile` 以使 Elastic Beanstalk 生成并运行自定义映像。此文件是可选的，具体取决于您的部署要求。有关 `Dockerfile` 的更多信息，请参阅 Docker 网站上的 Dockerfile 参考。
- 创建一个包含应用程序文件、应用程序文件依赖项、`.zip` 以及 `Dockerfile` 文件的 `docker-compose.yml` 文件。如果您使用 EB CLI 部署应用程序，它将为您创建一个 `.zip` 文件。这两个文件必须位于 `.zip` 存档的根级或顶级。

如果您只使用 `docker-compose.yml` 文件部署应用程序，则无需创建 `.zip` 文件。

本主题为语法参考。有关使用 Elastic Beanstalk 启动 Docker 环境的详细步骤，请参阅 使用 Docker 平台分支。

要了解有关 Docker Compose 以及如何安装它的更多信息，请参阅 Docker 网站 Docker Compose 概述和安装 Docker Compose。

> **Note**
>
> 如果您不使用 Docker Compose 配置 Docker 环境，也不应该使用 `docker-compose.yml` 文件。而是使用 `Dockerrun.aws.json` 文件和/或 `Dockerfile`。
> 有关更多信息，请参阅the section called “Docker 平台的配置（不含 Docker Compose）”。

使用私有存储库中的映像

Elastic Beanstalk 必须使用托管私有存储库的在线注册表进行身份验证，然后才能从私有存储库中提取和部署您的映像。我们提供了两个选项的示例，用于存储和检索 Elastic Beanstalk 环境的凭证，以便向存储库进行身份验证。

- 的 AWS Secrets Manager
- `Dockerrun.aws.json v3` 文件

使用 AWS Secrets Manager

您可以将 Elastic Beanstalk 配置为在它开始部署过程之前登录私有存储库。这样，Elastic Beanstalk 就可以从存储库访问映像并将这些映像部署到您的 Elastic Beanstalk 环境。

此配置在 Elastic Beanstalk 部署过程的预构建 阶段启动事件。您可以在 .ebextentions 配置目录中进行此设置。该配置使用平台挂钩脚本，这些脚本调用 docker login 对托管私有存储库的联机注册表进行身份验证。以下是这些配置步骤的详细分解说明。

使用 AWS Secrets Manager将 Elastic Beanstalk 配置为向私有存储库进行身份验证

> **Note**
>
> 必须授予特定权限才能完成这些步骤。有关更多信息，请参阅以下参考。

- 在步骤 2 中，您需要相应权限才能创建密钥。有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的示例：创建密钥的权限。
- 在步骤 3 中，您需要相应权限才能使用 secretsmanager 动态引用检索密钥。有关更多信息，请参阅AWS Secrets Manager 用户指南中的示例：检索密钥值的权限。

1. 按如下方式创建 .ebextensions 目录结构。

```
### .ebextensions
#   ### env.config
### .platform
#   ### confighooks
#   #   ### prebuild
#   #       ### 01login.sh
#   ### hooks
#       ### prebuild
#           ### 01login.sh
### docker-compose.yml
```

2. AWS Secrets Manager 用于保存您的私有存储库的证书，这样 Elastic Beanstalk 就可以在需要时检索您的证书。为此，请运行 Secrets Manager create-secre AWS CLI t 命令。

```
aws secretsmanager create-secret \
            --name MyTestSecret \
            --description "My image repo credentials created with the CLI." \
            --secret-string "{\"USER\":\"EXAMPLE-USERNAME\",\"PASSWD\":\"EXAMPLE-
PASSWORD\"}"
```

3. 创建以下 env.config 文件并将其放在 .ebextensions 目录中，如前面的目录结构所示。此配置使用 aws:elasticbeanstalk:application:environment 命名空间通过对 AWS Secrets Manager的动态引用初始化 USER 和 PASSWD Elastic Beanstalk 环境变量。有关secretsmanager动态引用的更多信息，请参阅《AWS Secrets Manager 用户指南》中的检索 AWS CloudFormation 资源中的 AWS Secrets Manager 密钥。

> **Note**
>
> 脚本中的 USER 和 PASSWD 必须与前面的 secretsmanager create-secret 命令中使用的相同字符串匹配。

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    USER: '{{resolve:secretsmanager:MyTestSecret:SecretString:USER}}'
    PASSWD: '{{resolve:secretsmanager:MyTestSecret:SecretString:PASSWD}}'
```

4. 创建以下 01login.sh 脚本文件并将其放在以下目录中（也显示在前面的目录结构中）：

   - .platform/confighooks/prebuild
   - .platform/hooks/prebuild

```
### example 01login.sh
#!/bin/bash
USER=/opt/elasticbeanstalk/bin/get-config environment -k USER
/opt/elasticbeanstalk/bin/get-config environment -k PASSWD | docker login -u $USER
 --password-stdin
```

01login.sh 脚本将调用 get-config 平台脚本来检索存储库凭证，然后登录到存储库。它将用户名存储在 USER 脚本变量中。在下一行中，它将检索密码。该脚本不将密码存储在脚本变量中，而是将密码直接传送给 stdin 输入流中的 docker login 命令。--password-stdin 选项使用输入流，因此您不必将密码存储在变量中。有关使用 Docker 命令行界面进行身份验证的更多信息，请参阅 Docker 文档网站上的 docker login。

> **注意**
>
> - 所有脚本文件都必须具有执行权限。使用 chmod +x 对挂钩文件设置执行权限。对于 2022 年 4 月 29 日或之后发布的所有基于 Amazon Linux 2 的平台版本，Elastic Beanstalk 会自动向所有平台挂钩脚本授予执行权限。在这种情况下，您无需手动授予执行权限。有关这些平台版本的列表，请参阅 AWS Elastic Beanstalk 发布说明指南中的 2022 年 4 月 29 日 - Linux 平台发布说明。
> - 挂钩文件既可以是二进制文件，也可以是以包含其解释器路径的 #! 行开头的脚本文件，例如 #!/bin/bash。
> - 有关更多信息，请参阅扩展 Elastic Beanstalk Linux 平台 中的 the section called “平台挂钩”。

在 Elastic Beanstalk 使用托管私有存储库的在线注册表进行身份验证之后，您可以提取和部署您的映像。

使用 Dockerrun.aws.json v3 文件

本节介绍向私有存储库验证 Elastic Beanstalk 的另一种方法。使用此方法，您可以使用 Docker 命令生成身份验证文件，然后将身份验证文件上传到 Amazon S3 存储桶。您还必须在 `Dockerrun.aws.json v3` 文件中包含存储桶信息。

生成身份验证文件并提供给 Elastic Beanstalk

1. 使用 docker login 命令生成身份验证文件。对于 Docker Hub 上的存储库，请运行 docker login：

   ```
   $ docker login
   ```

   对于其他注册表，请包括注册表服务器的 URL：

   ```
   $ docker login registry-server-url
   ```

   > **Note**
   >
   > 如果您的 Elastic Beanstalk 环境使用 Amazon Linux AMI Docker 平台版本（在 Amazon Linux 2 之前），请阅读the section called “Amazon Linux AMI（在 Amazon Linux 2 之前）上的 Docker 配置”中的相关信息。

   有关身份验证文件的更多信息，请参阅 Docker 网站上的在 Docker Hub 上存储映像和 docker login。

2. 将名为 `.dockercfg` 的身份验证文件的副本上传到安全的 Amazon S3 存储桶。

   - Amazon S3 存储桶的托管环境必须与使用它的环境 AWS 区域 相同。Elastic Beanstalk 无法从托管在其他区域的 Amazon S3 存储桶下载文件。
   - 在实例配置文件中授予 IAM 角色执行 `s3:GetObject` 操作的权限。有关更多信息，请参阅管理 Elastic Beanstalk 实例配置文件。

3. 在 `Authentication` 文件的 `Dockerrun.aws.json v3` 参数中包含 Amazon S3 存储桶信息。

   以下是 `Dockerrun.aws.json v3` 文件的示例。

```
{
  "AWSEBDockerrunVersion": "3",
  "Authentication": {
    "bucket": "DOC-EXAMPLE-BUCKET",
    "key": "mydockercfg"
  }
}
```

> **Note**
>
> AWSEBDockerrunVersion 参数指示 Dockerrun.aws.json 文件的版本。
>
> - Docker Amazon Linux 2 平台将 Dockerrun.aws.json v3 文件用于使用 Docker Compose 的环境。它将 Dockerrun.aws.json v1 文件用于不使用 Docker Compose 的环境。
> - 多容器 Docker Amazon Linux AMI 平台使用 Dockerrun.aws.json v2 文件。

在 Elastic Beanstalk 可以使用托管私有存储库的在线注册表进行身份验证之后，可以部署和提取您的映像。

使用 Dockerfile 构建自定义映像

如果您尚未在存储库中托管现有映像，则需要创建 Dockerfile。

以下代码段是一个 Dockerfile 示例。如果您按照 使用 Docker 平台分支 中的说明进行操作，则可以按编写内容上传此 Dockerfile。在您使用此 Dockerfile 时，Elastic Beanstalk 运行游戏 2048。

有关您可以包含在 Dockerfile 中的指令的更多信息，请参阅 Docker 网站上的 Dockerfile 参考。

```
FROM ubuntu:12.04

RUN apt-get update
RUN apt-get install -y nginx zip curl

RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN curl -o /usr/share/nginx/www/master.zip -L https://codeload.github.com/
gabrielecirulli/2048/zip/master
RUN cd /usr/share/nginx/www/ && unzip master.zip && mv 2048-master/* . && rm -rf 2048-
master master.zip
```

```
EXPOSE 80

CMD ["/usr/sbin/nginx", "-c", "/etc/nginx/nginx.conf"]
```

> **Note**
>
> 您可以从单个 Dockerfile 运行多阶段构建来生成较小尺寸的映像，同时显著降低复杂性。有关更多信息，请参阅 Docker 文档网站上的使用多阶段构建。

Docker 平台的配置（不含 Docker Compose）

如果您的 Elastic Beanstalk Docker 环境不使用 Docker Compose，请阅读以下部分中的其他信息。

Docker 平台配置 - 不含 Docker Compose

您部署到 Docker 环境中的 Elastic Beanstalk 的任何 Web 应用程序都必须包含 `Dockerfile` 或 `Dockerrun.aws.json` 文件。您可以通过执行以下操作之一将 Web 应用程序从 Docker 容器部署到 Elastic Beanstalk：

- 创建 `Dockerfile` 以使 Elastic Beanstalk 生成并运行自定义映像。
- 创建一个用于将 Docker 映像从托管存储库部署到 Elastic Beanstalk 的 `Dockerrun.aws.json` 文件。
- 创建一个包含应用程序文件、应用程序文件依赖项、`.zip` 以及 `Dockerfile` 文件的 `Dockerrun.aws.json` 文件。如果您使用 EB CLI 部署应用程序，它将为您创建一个 `.zip` 文件。

  如果您只用一个 `Dockerfile` 或只用一个 `Dockerrun.aws.json` 文件部署应用程序，则无需创建 `.zip` 文件。

本主题为语法参考。有关启动 Docker 环境的详细步骤，请参阅 使用 Docker 平台分支。

`Dockerrun.aws.json` v1

`Dockerrun.aws.json` 文件描述如何将远程 Docker 映像部署为 Elastic Beanstalk 应用程序。此 JSON 文件特定于 Elastic Beanstalk。如果应用程序在托管存储库中提供的映像上运行，您可以在 `Dockerrun.aws.json` v1 文件中指定该映像并忽略 `Dockerfile`。

`Dockerrun.aws.json` v1 文件的有效键和值包括以下操作：

AWSEBDockerrunVersion

(必需) 为单容器 Docker 环境将版本号指定为值 1。

身份验证

（仅对私有存储库必需）指定存储 `.dockercfg` 文件的 Amazon S3 对象。

请参阅 使用私有存储库中的映像。

映像

指定现有 Docker 存储库上的 Docker 基本映像，您将从其构建 Docker 容器。指定 Name 键的值：对于 Docker Hub 上的映像，采用 *<organization>/<image name>* 的格式；对于其他站点，采用 *<site>/<organization name>/<image name>* 的格式。

在 `Dockerrun.aws.json` 文件中指定一个映像后，您的 Elastic Beanstalk 环境中的每个实例都运行 `docker pull` 来运行该映像。可以选择包含 Update 键。默认值为 `true`，指示 Elastic Beanstalk 检查存储库，提取映像的所有更新并覆盖任何缓存的映像。

使用 `Dockerfile` 时，请勿在 `Dockerrun.aws.json` 文件中指定 Image 键。当 `Dockerfile` 中所述的映像存在时，Elastic Beanstalk 始终构建并使用该映像。

端口

（指定 Image 键时必需）列出要在 Docker 容器上公开的端口。Elastic Beanstalk ContainerPort使用该值将 Docker 容器连接到主机上运行的反向代理。

您可以指定多个容器端口，但 Elastic Beanstalk 只使用第一个端口。它只使用此端口将您的容器连接到主机的反向代理并路由来自公有 Internet 的请求。如果您使用的是**`Dockerfile`**，则第一个ContainerPort值应与的 EXPOSE 列表中的**`Dockerfile`**第一个条目相匹配。

或者，您也可以在中指定端口列表HostPort。HostPort条目指定ContainerPort值映射到的主机端口。如果未指定HostPort值，则默认为该ContainerPort值。

```
{
  "Image": {
    "Name": "image-name"
  },
  "Ports": [
    {
      "ContainerPort": 8080,
      "HostPort": 8000
    }
```

```
  ]
}
```

卷

将卷从 EC2 实例映射到 Docker 容器。指定要映射的一个或多个卷数组。

```
{
  "Volumes": [
    {
      "HostDirectory": "/path/inside/host",
      "ContainerDirectory": "/path/inside/container"
    }
  ]
...
```

日志记录

指定应用程序将日志写入到的容器内目录。当您请求结尾日志或捆绑日志时，Elastic Beanstalk 会将此目录中的所有日志上传到 Amazon S3。如果您将日志轮换到此目录中名为 `rotated` 的文件夹，则还可以将 Elastic Beanstalk 配置为将轮换的日志上传到 Amazon S3 以进行永久存储。有关更多信息，请参阅查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志。

命令

指定要在容器中运行的命令。如果您指定 Entrypoint，Command 将作为参数添加至 Entrypoint。有关更多信息，请参阅 Docker 文档中的 CMD。

Entrypoint

指定要在容器启动时运行的默认命令。有关更多信息，请参阅 Docker 文档中的 ENTRYPOINT。

以下代码段是演示单容器的 `Dockerrun.aws.json` 文件的语法的一个示例。

```
{
  "AWSEBDockerrunVersion": "1",
  "Image": {
    "Name": "janedoe/image",
    "Update": "true"
```

```
  },
  "Ports": [
    {
      "ContainerPort": "1234"
    }
  ],
  "Volumes": [
    {
      "HostDirectory": "/var/app/mydb",
      "ContainerDirectory": "/etc/mysql"
    }
  ],
  "Logging": "/var/log/nginx",
  "Entrypoint": "/app/bin/myapp",
  "Command": "--argument"
}
```

您可以仅向 Elastic Beanstalk 提供 Dockerrun.aws.json 文件，也可以提供包含 .zip 和 Dockerrun.aws.json 文件的 Dockerfile 存档。在提供这两个文件时，Dockerfile 描述 Docker 映像，Dockerrun.aws.json 文件提供其他部署信息，如本节后面所述。

> **Note**
>
> 这两个文件必须位于 .zip 存档的根级或顶级。请不要从包含这些文件的目录生成存档。而是导航到该目录并在其中生成存档。
>
> 在提供这两个文件时，请不要在 Dockerrun.aws.json 文件中指定映像。Elastic Beanstalk 生成和使用 Dockerfile 中描述的映像，并忽略 Dockerrun.aws.json 文件中指定的映像。

使用私有存储库中的映像

在 Authentication 文件的 Dockerrun.aws.json v1 参数中添加有关包含身份验证文件的 Amazon S3 存储桶的信息。确保 Authentication 参数包含有效的 Amazon S3 存储桶和密钥。Amazon S3 存储桶必须托管在使用它的环境所在的相同 AWS 区域 中。Elastic Beanstalk 不会从托管在其他区域的 Amazon S3 存储桶下载文件。

有关生成和上传身份验证文件的信息，请参阅 使用私有存储库中的映像。

下面的示例演示了 DOC-EXAMPLE-BUCKET 存储桶中一个名为 mydockercfg 的身份验证文件如何使用第三方注册表中的私有映像。

```
{
  "AWSEBDockerrunVersion": "1",
  "Authentication": {
    "Bucket": "DOC-EXAMPLE-BUCKET",
    "Key": "mydockercfg"
  },
  "Image": {
    "Name": "quay.io/johndoe/private-image",
    "Update": "true"
  },
  "Ports": [
    {
      "ContainerPort": "1234"
    }
  ],
  "Volumes": [
    {
      "HostDirectory": "/var/app/mydb",
      "ContainerDirectory": "/etc/mysql"
    }
  ],
  "Logging": "/var/log/nginx"
}
```

配置 Docker 环境

有几种方法可以配置 Elastic Beanstalk Docker 环境的行为。

> **Note**
>
> 如果您的 Elastic Beanstalk 环境使用 Amazon Linux AMI Docker 平台版本（在 Amazon Linux 2 之前），请务必阅读 the section called “Amazon Linux AMI（在 Amazon Linux 2 之前）上的 Docker 配置”中的其他信息。

小节目录

- Docker 容器自定义日志记录 (Docker Compose)
- Docker 映像
- 配置 Docker 环境的托管更新
- Docker 配置命名空间
- Amazon Linux AMI（在 Amazon Linux 2 之前）上的 Docker 配置

在 Docker 环境中配置软件

您可以使用 Elastic Beanstalk 控制台来配置在您的环境实例上运行的软件。

在 Elastic Beanstalk 控制台中配置 Docker 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 进行必要的配置更改。
6. 要保存更改，请选择页面底部的 Apply（应用）。

有关在任何环境中配置软件设置的信息，请参阅the section called “环境属性和软件设置”。以下各部分介绍特定于 Docker 的信息。

容器选项

容器选项部分具有特定于平台的选项。对于 Docker 环境，它允许您选择您的环境是否包含 NGINX 代理服务器。

使用 Docker Compose 的环境

如果您使用 Docker Compose 管理 Docker 环境，Elastic Beanstalk 假定您将代理服务器作为容器运行。因此，对于代理服务器设置，它默认为无，Elastic Beanstalk 不提供 NGINX 配置。

> **Note**
>
> 即使您选择 NGINX 作为代理服务器，也会在使用 Docker Compose 的环境中忽略此设置。代理服务器设置仍默认为无。

由于对于使用 Docker Compose 的 Amazon Linux 2 平台上的 Docker 已禁用 NGINX Web 服务器代理，因此您必须按照为增强型运行状况报告生成日志的说明操作。有关更多信息，请参阅为增强型运行状况报告生成日志 (Docker Compose)。

环境属性和环境变量

在 Environment properties (环境属性) 部分，您可以在运行应用程序的 Amazon Elastic Compute Cloud (Amazon EC2) 实例上指定环境配置设置。环境属性会以密钥值对的形式传递到应用程序。在 Docker 环境中，Elastic Beanstalk 将环境属性作为环境变量传递给容器。

在容器中运行的应用程序代码可以按名称引用环境变量并读取其值。读取这些环境变量的源代码将因语言而异。您可以在相应的平台主题中找到采用 Elastic Beanstalk 托管平台支持的编程语言读取环境变量值的说明。有关指向这些主题的链接的列表，请参阅the section called “环境属性和软件设置”。

使用 Docker Compose 的环境

如果您使用 Docker Compose 管理 Docker 环境，则必须进行一些额外的配置以检索容器中的环境变量。为了使容器中运行的可执行文件能够访问这些环境变量，您必须在 `docker-compose.yml` 中引用它们。有关更多信息，请参阅在容器中引用环境变量。

在容器中引用环境变量

如果您在 Amazon Linux 2 Docker 平台上使用 Docker Compose 工具，则 Elastic Beanstalk 会在应用程序项目的根目录中生成一个名为 `.env` 的 Docker Compose 环境文件。此文件存储您为 Elastic Beanstalk 配置的环境变量。

> **Note**
>
> 如果您在应用程序包中包含 `.env` 文件，则 Elastic Beanstalk 将不会生成 `.env` 文件。

为了让容器引用您在 Elastic Beanstalk 中定义的环境变量，必须遵循以下一种或两种配置方法。

- 将 Elastic Beanstalk 生成的 `.env` 文件添加到 `env_file` 文件的 `docker-compose.yml` 配置选项中。

- 直接在 docker-compose.yml 文件中定义环境变量。

以下文件提供了一个示例。示例 docker-compose.yml 文件演示了这两种方法。

- 如果您定义环境属性 DEBUG_LEVEL=1 和 LOG_LEVEL=error，则 Elastic Beanstalk 会为您生成以下 .env 文件：

```
DEBUG_LEVEL=1
LOG_LEVEL=error
```

- 在此 docker-compose.yml 文件中，env_file 配置选项指向 .env 文件，并且它还直接在 DEBUG=1 文件中定义环境变量 docker-compose.yml。

```
services:
  web:
    build: .
    environment:
      - DEBUG=1
    env_file:
      - .env
```

> **注意**
>
> - 如果在两个文件中设置相同的环境变量，则在 docker-compose.yml 文件中定义的变量的优先级高于在 .env 文件中定义的变量。
> - 注意不要在等号 (=) 和分配给变量的值之间留下空格，以防止将空格添加到字符串中。

要了解有关 Docker Compose 中环境变量的更多信息，请参阅 Compose 中的环境变量

对环境变量使用插值功能（Docker Compose）

Docker Amazon Linux 2 平台分支从 2023 年 7 月 28 日平台版本发布起，提供 Docker Compose 插值功能。借助此功能，可通过变量设置 Compose 文件中的值，并在运行时进行插值。有关此功能的更多信息，请参阅 Docker 文档网站上的插值。

> **Important**
>
> 如果您想在应用程序中使用此功能，请注意，您需要应用一种使用平台挂钩的方法。必须执行此操作，因为我们已在平台引擎中执行缓解操作。该缓解操作可确保不了解新插值功能且现有应用程序使用带 $ 字符环境变量的客户实现向后兼容。默认情况下，更新后的平台引擎通过将 $ 字符替换为 $$ 字符对插值进行转义。

以下是平台挂钩脚本示例，您可以将其设置为允许使用插值功能。

```
#!/bin/bash

: '
example data format in .env file
key1=value1
key2=value2
'
envfile="/var/app/staging/.env"
tempfile=$(mktemp)

while IFS= read -r line; do
  # split each env var string at '='
  split_str=(${line//=/ })
  if [ ${#split_str[@]} -eq 2 ]; then
    # replace '$$' with '$'
    replaced_str=${split_str[1]//\$\$/\$}
    # update the value of env var using ${replaced_str}
    line="${split_str[0]}=${replaced_str}"
  fi
  # append the updated env var to the tempfile
  echo "${line}" #"${tempfile}"
done < "${envfile}"
# replace the original .env file with the tempfile
mv "${tempfile}" "${envfile}"
```

将平台挂钩放在以下两个目录下：

- `.platform/confighooks/predeploy/`
- `.platform/hooks/predeploy/`

有关更多信息，请参阅本指南中扩展 Linux 平台主题中的 平台挂钩。

为增强型运行状况报告生成日志 (Docker Compose)

Elastic Beanstalk 运行状况代理为 Elastic Beanstalk 环境提供操作系统和应用程序运行状况指标。它依赖于以特定格式中继信息的 Web 服务器日志格式。

Elastic Beanstalk 假设您将 Web 服务器代理作为容器运行。因此，对于运行 Docker Compose 的 Docker 环境禁用 NGINX Web 服务器代理。您必须将服务器配置为以 Elastic Beanstalk 运行状况代理所使用的位置和格式写入日志。这样就可以充分利用增强型运行状况报告，即使禁用了 Web 服务器代理也是如此。

有关如何执行此操作的说明，请参阅Web 服务器日志配置

Docker 容器自定义日志记录 (Docker Compose)

为了有效地排查问题并监控容器化服务，您可以通过环境管理控制台或 EB CLI 从 Elastic Beanstalk 请求实例日志。实例日志由服务包日志和结尾日志组成，可组合并打包以便您能够以高效、直接的方式查看日志和最近事件。

Elastic Beanstalk 在容器实例上创建日志目录，在 `docker-compose.yml` 处为 `/var/log/eb-docker/containers/<service name>` 文件中定义的每个服务创建一个日志目录。如果您在 Amazon Linux 2 Docker 平台上使用 Docker Compose 功能，则可以将这些目录挂载到容器文件结构中写入日志的位置。当您挂载日志目录以写入日志数据时，Elastic Beanstalk 可以从这些目录中收集日志数据。

如果您的应用程序位于不使用 Docker Compose 的 Docker 平台上，则可以按照 Docker 容器自定义日志记录 (Docker Compose)中所述的标准过程进行操作。

将服务的日志文件配置为可撤回的结尾文件和服务包日志

1. 编辑 `docker-compose.yml` 文件。
2. 在服务的 `volumes` 键下添加绑定挂载，如下所示：

   `"${EB_LOG_BASE_DIR}/<service name>:<log directory inside container>`

   在下面的示例 `docker-compose.yml` 文件中：

   - `nginx-proxy` 是 `<####>`
   - `/var/log/nginx` 是 `<########>`

```
services:
  nginx-proxy:
    image: "nginx"
    volumes:
      - "${EB_LOG_BASE_DIR}/nginx-proxy:/var/log/nginx"
```

- var/log/nginx 目录包含容器中 nginx-proxy 服务的日志，它将映射到主机上的 /var/log/eb-docker/containers/nginx-proxy 目录。
- 此目录中的所有日志现在都可以通过 Elastic Beanstalk 的请求实例日志功能作为捆绑日志和结尾日志进行检索。

> **注意**
>
> - ${EB_LOG_BASE_DIR} 是一个由 Elastic Beanstalk 设置的环境变量，值为 /var/log/eb-docker/containers。
> - Elastic Beanstalk 会自动为 /var/log/eb-docker/containers/*<service name>* 文件中的每个服务创建 docker-compose.yml 目录。

Docker 映像

Elastic Beanstalk 的 Docker 和 ECS 托管式 Docker 平台分支，可支持使用公有或私有线上镜像存储库中存储的 Docker 镜像。

在 Dockerrun.aws.json 中按名称指定映像。记下这些约定：

- Docker Hub 上的官方存储库中的映像使用一个名称 (例如，ubuntu 或 mongo)。
- Docker Hub 上其他存储库中的映像通过组织名称 (例如，amazon/amazon-ecs-agent) 进行限定。
- 其他在线存储库中的映像由域名 (例如，quay.io/assemblyline/ubuntu 或 *account-id*.dkr.ecr.us-east-2.amazonaws.com/ubuntu:trusty) 进行进一步限定。

对于仅使用 Docker 平台的环境，您还可以在创建环境期间使用 Dockerfile 构建自己的映像。有关更多信息，请参阅 使用 Dockerfile 构建自定义映像。多容器 Docker 平台不支持此功能。

使用 Amazon ECR 存储库中的映像

您可以使用亚马逊弹性容器注册表 (Amazon ECR) 存储您的自定义 Docker 镜像。 AWS 当您将 Docker 映像存储在 Amazon ECR 中时，Elastic Beanstalk 会自动使用环境的实例配置文件向 Amazon ECR 注册表进行身份验证，因此您无需生成身份验证文件并将其上传到 Amazon Simple Storage Service (Amazon S3)。

但您需要为实例提供访问 Amazon ECR 存储库中的镜像的权限 (通过将权限添加到环境的实例配置文件中)。您可以将 AmazonEC2 ContainerRegistryReadOnly 托管策略附加到实例配置文件以提供对您账户中所有 Amazon ECR 存储库的只读访问权限，或者使用以下模板创建自定义策略来授予对单个存储库的访问权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "AllowEbAuth",
        "Effect": "Allow",
        "Action": [
          "ecr:GetAuthorizationToken"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Sid": "AllowPull",
        "Effect": "Allow",
        "Resource": [
          "arn:aws:ecr:us-east-2:account-id:repository/repository-name"
        ],
        "Action": [
          "ecr:GetAuthorizationToken",
          "ecr:BatchCheckLayerAvailability",
          "ecr:GetDownloadUrlForLayer",
          "ecr:GetRepositoryPolicy",
          "ecr:DescribeRepositories",
          "ecr:ListImages",
          "ecr:BatchGetImage"
        ]
```

```
      }
    ]
  }
```

将上述策略中的 Amazon Resource Name (ARN) 替换为存储库的 ARN。

在 Dockerrun.aws.json 文件中，按 URL 引用镜像。对于 Docker 平台，URL 进入 Image 定义：

```
  "Image": {
      "Name": "account-id.dkr.ecr.us-east-2.amazonaws.com/repository-name:latest",
      "Update": "true"
    },
```

对于多容器 Docker 平台，请在容器定义对象中使用 image 键：

```
"containerDefinitions": [
        {
        "name": "my-image",
        "image": "account-id.dkr.ecr.us-east-2.amazonaws.com/repository-name:latest",
```

使用私有存储库中的映像

要使用由在线注册表托管的私有存储库中的 Docker 映像，必须提供一个身份验证文件，其中含有向注册表进行身份验证所需的信息。

使用 docker login 命令生成身份验证文件。对于 Docker Hub 上的存储库，请运行 docker login：

```
$ docker login
```

对于其他注册表，请包括注册表服务器的 URL：

```
$ docker login registry-server-url
```

> **Note**
>
> 如果您的 Elastic Beanstalk 环境使用 Amazon Linux AMI Docker 平台版本（在 Amazon Linux 2 之前），请阅读 the section called “Amazon Linux AMI（在 Amazon Linux 2 之前）上的 Docker 配置”中的其他信息。

将名为 `.dockercfg` 的身份验证文件副本上传到安全的 Amazon S3 存储桶。Amazon S3 存储桶必须托管在与使用它的环境相同的 AWS 区域。Elastic Beanstalk 无法从托管在其他区域的 Amazon S3 存储桶下载文件。在实例配置文件中授予 IAM 角色执行 `s3:GetObject` 操作的权限。有关详细信息，请参阅管理 Elastic Beanstalk 实例配置文件。

在 `Authentication` 文件的 `authentication` (v1) 或 `Dockerrun.aws.json` (v2) 参数中包含 Amazon S3 存储桶信息。

有关 Docker 环境的 `Dockerrun.aws.json` 格式的更多信息，请参阅 Docker 配置。对于多容器环境，请参阅 ECS 托管式 Docker 配置。

有关身份验证文件的更多信息，请参阅 Docker 网站上的在 Docker Hub 上存储映像和 docker login。

配置 Docker 环境的托管更新

利用托管平台更新，您可以将环境配置为按计划自动更新为平台的最新版本。

在使用 Docker 环境时，您可能希望确定在新平台版本包括新 Docker 版本时，是否应跨 Docker 版本进行自动平台更新。从运行高于 2.9.0 的 Docker 平台版本的环境中进行更新时，Elastic Beanstalk 支持跨 Docker 版本的托管平台更新。当新平台版本包含 Docker 的新版本时，Elastic Beanstalk 会递增次要更新版本号。因此，要允许托管平台跨 Docker 版本更新，请为托管平台同时启用次版本更新和修补版本更新。要防止托管平台跨 Docker 版本更新，请为托管平台更新启用仅应用修补版本更新。

例如，以下配置文件在每个星期二上午 9:00 (UTC) 启用次版本更新和修补版本更新，从而允许跨 Docker 版本的托管更新：

Example .ebextensions/ .config managed-platform-update

```
option_settings:
  aws:elasticbeanstalk:managedactions:
    ManagedActionsEnabled: true
    PreferredStartTime: "Tue:09:00"
  aws:elasticbeanstalk:managedactions:platformupdate:
    UpdateLevel: minor
```

对于运行 Docker 平台版本 2.9.0 或更早版本的环境，在新平台版本包括新 Docker 版本时，Elastic Beanstalk 从不执行托管平台更新。

Docker 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

> **Note**
>
> 此信息仅适用于未运行 Docker Compose 的 Docker 环境。此选项在运行 Docker Compose 的 Docker 环境中具有不同的行为。有关使用 Docker Compose 的代理服务的更多信息，请参阅容器选项。

除了所有 Elastic Beanstalk 环境支持的选项外，Docker 平台还支持以下命名空间中的选项：

- `aws:elasticbeanstalk:environment:proxy` – 为您的环境选择代理服务器。Docker 支持运行 Nginx 或没有代理服务器。

以下示例配置文件将 Docker 环境配置为不运行代理服务器。

Example .ebextensions/docker-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: none
```

Amazon Linux AMI（在 Amazon Linux 2 之前）上的 Docker 配置

如果您的 Elastic Beanstalk Docker 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

将身份验证文件用于私有存储库

如果您使用私有存储库中的映像，则此信息对您有用。从 Docker 1.7 版开始，docker login 命令更改了身份验证文件的名称和格式。Amazon Linux AMI Docker 平台版本（在 Amazon Linux 2 之前）需要较旧的 `~/.dockercfg` 格式配置文件。

对于 Docker 1.7 版和更高版本，docker login 命令采用以下格式在 `~/.docker/config.json` 中创建身份验证文件。

```
{
    "auths":{
      "server":{
        "auth":"key"
      }
    }
```

```
  }
```

对于 Docker 1.6.2 版和更早版本，docker login 命令采用以下格式在 ~/.dockercfg 中创建身份验证文件。

```
{
    "server" :
    {
      "auth" : "auth_token",
      "email" : "email"
    }
  }
```

要转换 config.json 文件，请删除外部 auths 键、添加 email 键，并平展 JSON 文档以匹配旧格式。

在 Amazon Linux 2 Docker 平台版本上，Elastic Beanstalk 使用较新的身份验证文件名和格式。如果您使用的是 Amazon Linux 2 Docker 平台版本，则可以在不进行任何转换的情况下使用 docker login 命令创建的身份验证文件。

配置额外的存储卷

为了提高 Amazon Linux AMI 的性能，Elastic Beanstalk 会为 Docker 环境的 Amazon EC2 实例配置两个 Amazon EBS 存储卷。除了为所有 Elastic Beanstalk 环境预配置的根卷之外，还为 Docker 环境中的映像存储预配置了第二个 12GB 的卷（名为 xvdcz）。

如果您需要为 Docker 映像增加存储空间或者提高 IOPS，可以使用 aws:autoscaling:launchconfiguration 命名空间中的 BlockDeviceMapping 配置选项来自定义映像存储卷。

例如，以下配置文件将存储卷大小增加到 100 GB，预配置的 IOPS 为 500：

Example .ebextensions/blockdevice-xvdcz.config

```
option_settings:
  aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: /dev/xvdcz=:100::io1:500
```

如果您使用 BlockDeviceMappings 选项为应用程序配置附加卷，那么您应该包括 xvdcz 的映射以确保创建了该卷。以下示例配置两个卷，一个是具有默认设置的映像存储卷 xvdcz，另一个是名为 sdh 的 24 GB 应用程序卷：

Example .ebextensions/blockdevice-sdh.config

```
option_settings:
  aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: /dev/xvdcz=:12:true:gp2,/dev/sdh=:24
```

> **Note**
>
> 当您更改此命名空间中的设置时，Elastic Beanstalk 会将环境中的所有实例替换为运行新配置的实例。有关详细信息，请参阅 配置更改。

使用 Amazon ECS 平台分支

本主题介绍了在 Amazon Linux 2 上运行的 Amazon ECS 平台分支，以及它替换的平台分支，即，AL1 上的多容器 Docker（也由 ECS 托管）。除非另有说明，本主题中的所有信息均适用于这两个平台分支。

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。

从 AL1 上的多容器 Docker 迁移

如果您目前在使用已停用的在 AL1 上运行的多容器 Docker，则可以迁移到最新的在 AL2023 上运行的 ECS 平台分支。最新的平台分支支持已停产平台分支的所有功能。源代码无需更改。有关更多信息，请参阅将在 Amazon Linux 上运行的多容器 Docker 迁移到 Amazon Linux 2023 上的 ECS。

主题

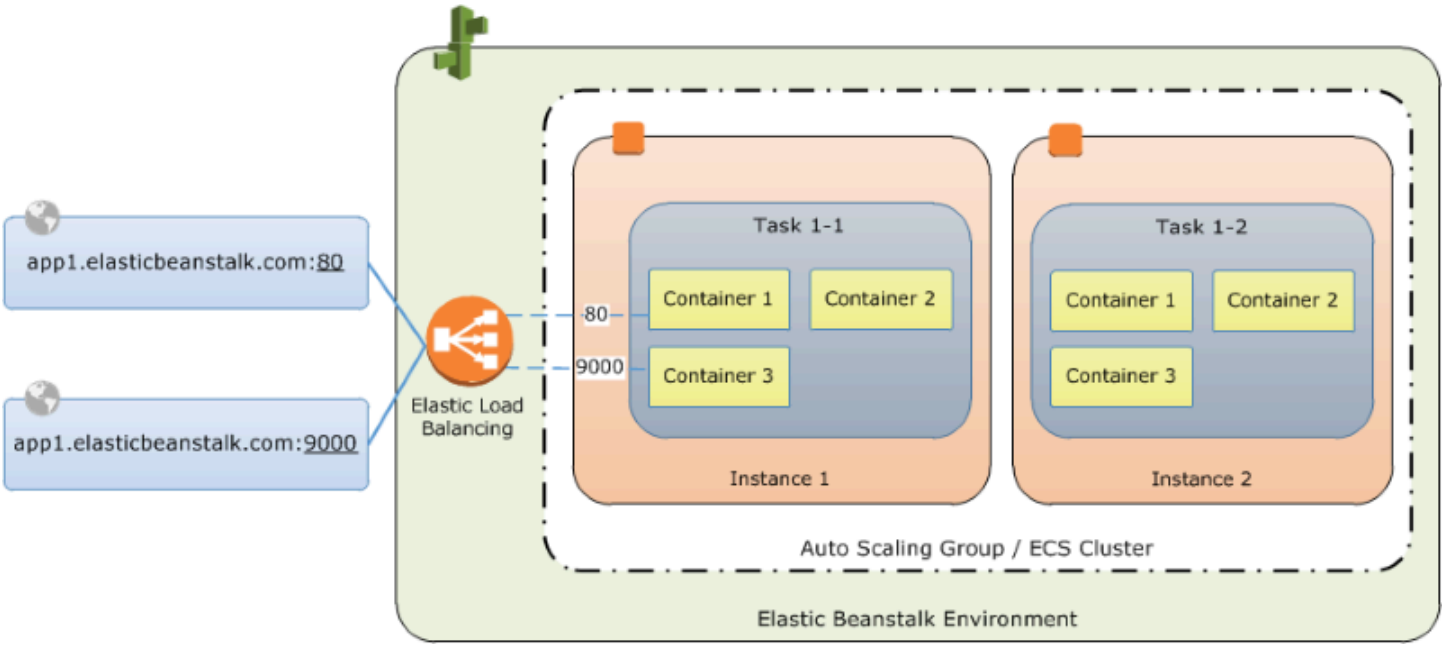
- ECS 托管的 Docker 平台
- Dockerrun.aws.json file
- Docker 映像
- 容器实例角色
- 由 Elastic Beanstalk 创建的 Amazon ECS 资源

- 使用多个 Elastic Load Balancing 侦听器
- 失败的容器部署
- ECS 托管式 Docker 配置
- 使用 Elastic Beanstalk 控制台启动 ECS 托管式 Docker 环境
- 将在 Amazon Linux 上运行的多容器 Docker 迁移到 Amazon Linux 2023 上的 ECS
- （旧式）从在 Amazon Linux 上运行的多容器 Docker 迁移到在 Amazon Linux 2 上运行的 Docker 平台分支

ECS 托管的 Docker 平台

Elastic Beanstalk 使用 Amazon Elastic Container Service (Amazon ECS) 来协调 ECS 托管式 Docker 环境中的容器部署。Amazon ECS 提供工具来管理运行 Docker 容器的实例的集群。Elastic Beanstalk 处理 Amazon ECS 任务，包括集群创建、任务定义和执行。环境中的每个实例都运行相同的容器组，如 `Dockerrun.aws.json` v2 文件所定义。为了充分利用 Docker，Elastic Beanstalk 允许您创建 Amazon EC2 实例可在其中并排运行多个 Docker 容器的环境。

下图显示了一个示例 Elastic Beanstalk 环境，该环境配置有三个在 Auto Scaling 组中的每个 Amazon EC2 实例上运行的 Docker 容器：



> **Note**
>
> Elastic Beanstalk 为其所有平台提供了可扩展性功能，以用来自定义应用程序的部署和运行。对于在 Amazon Linux 2 上运行的 ECS 平台分支，这些功能的实例部署工作流实现与其他平台

有所不同。有关更多信息，请参阅在 Amazon Linux 2 及更高版本上运行的 ECS 的实例部署工作流。

`Dockerrun.aws.json` file

容器实例 – 在 Elastic Beanstalk 环境中运行 ECS 托管式 Docker 的 Amazon EC2 实例 – 需要名为 `Dockerrun.aws.json` 的配置文件。此文件特定于 Elastic Beanstalk，可以单独使用，也可以与源包中的源代码和内容结合使用，以在 Docker 平台上创建环境。

> **Note**
>
> `Dockerrun.aws.json` 格式的版本 1 用于将单个 Docker 容器启动到 Amazon Linux AMI（Amazon Linux 2 之前的版本）上运行的 Elastic Beanstalk 环境。该环境基于 64 位 Amazon Linux 上运行的 Docker 平台分支（将于 2022 年 7 月 18 日停用）。要了解有关 `Dockerrun.aws.json` v1 格式的更多信息，请参阅 Docker 平台配置 - 不含 Docker Compose。
>
> `Dockerrun.aws.json` 版本 2 格式增加了对每个 Amazon EC2 实例对应多个容器的支持，并且只能与 ECS 托管的 Docker 平台一起使用。格式与上一个版本存在很大的差异。

有关更新的格式和示例文件的详细信息，请参阅`Dockerrun.aws.json` v2。

Docker 映像

适用于 Elastic Beanstalk 的 ECS 托管式 Docker 平台需要预构建映像并将其存储在公有或私有的在线映像存储库中。

> **Note**
>
> Elastic Beanstalk 上的 ECS 托管式 Docker 平台不支持使用 `Dockerfile` 在部署期间构建自定义映像。在创建 Elastic Beanstalk 环境之前，先构建映像并将其部署到在线存储库中。

在 `Dockerrun.aws.json` v2 中按名称指定映像。记下这些约定：

- Docker Hub 上的官方存储库中的映像使用一个名称 (例如，`ubuntu` 或 `mongo`)。
- Docker Hub 上其他存储库中的映像通过组织名称 (例如，`amazon/amazon-ecs-agent`) 进行限定。

- 其他在线注册表中的映像由域名 (例如，quay.io/assemblyline/ubuntu) 进行进一步限定。

要将 Elastic Beanstalk 配置为向私有存储库进行身份验证，请在 authentication v2 文件中包含 Dockerrun.aws.json 参数。

容器实例角色

Elastic Beanstalk 将 Amazon ECS 优化的 AMI 与在 Docker 容器中运行的 Amazon ECS 容器代理结合使用。该代理与 Amazon ECS 通信以协调容器部署。为了与 Amazon ECS 通信，每个 Amazon EC2 实例都必须在 IAM 中具有相应的权限。当您在 Elastic Beanstalk 控制台中创建环境时，这些权限将附加到默认的实例配置文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ECSAccess",
      "Effect": "Allow",
      "Action": [
        "ecs:Poll",
        "ecs:StartTask",
        "ecs:StopTask",
        "ecs:DiscoverPollEndpoint",
        "ecs:StartTelemetrySession",
        "ecs:RegisterContainerInstance",
        "ecs:DeregisterContainerInstance",
        "ecs:DescribeContainerInstances",
        "ecs:Submit*"
      ],
      "Resource": "*"
    }
  ]
}
```

如果您自行创建实例配置文件，可连接 AWSElasticBeanstalkMulticontainerDocker 托管策略，以确定许可保持最新。有关在 IAM 中创建策略和角色的说明，请参阅《IAM 用户指南》中的创建 IAM 角色。

由 Elastic Beanstalk 创建的 Amazon ECS 资源

当您使用 ECS 托管的 Docker 平台创建环境时，Elastic Beanstalk 会在构建环境的同时自动创建并配置多个 Amazon Elastic Container Service 资源。在此过程中，它会在每个 Amazon EC2 实例上创建所需的容器。

- Amazon ECS 集群 – Amazon ECS 中的容器实例被组成集群。与 Elastic Beanstalk 一起使用时，始终为每个 ECS 托管式 Docker 环境创建一个集群。
- Amazon ECS 任务定义 – Elastic Beanstalk 使用您项目中的 `Dockerrun.aws.json` v2 生成用于在环境中配置容器实例的 Amazon ECS 任务定义。
- Amazon ECS 任务 – Elastic Beanstalk 与 Amazon ECS 进行通信以在环境中的每个实例上运行一个任务，从而协调容器部署。在可扩展的环境中，只要将一个实例添加到集群中，Elastic Beanstalk 就会启动一个新任务。在极少数情况下，您可能需要增加为容器和映像预留的空间量。请阅读配置 Docker 环境部分了解更多信息。
- Amazon ECS 容器代理 – 代理在环境中实例上的 Docker 容器中运行。该代理轮询 Amazon ECS 服务并等待任务运行。
- Amazon ECS 数据卷 – Elastic Beanstalk 在任务定义中插入卷定义（除了您在 `Dockerrun.aws.json` v2 中定义的卷之外），以方便日志收集。

  Elastic Beanstalk 在 `/var/log/containers/`*`containername`* 处的容器实例上创建日志卷（为每个容器创建一个日志卷）。这些卷将命名为 `awseb-logs-`*`containername`* 并提供给要挂载的容器。请参阅 容器定义格式 以了解有关如何装载容器的详细信息。

使用多个 Elastic Load Balancing 侦听器

您可以在 ECS 托管式 Docker 环境中配置多个 Elastic Load Balancing 侦听器，以支持不在默认 HTTP 端口上运行的代理或其他服务的入站流量。

在您的源包中创建一个 `.ebextensions` 文件夹并添加一个文件扩展名为 `.config` 的文件。以下示例显示了用于在端口 8080 上创建 Elastic Load Balancing 侦听器的配置文件。

`.ebextensions/elb-listener.config`

```
option_settings:
  aws:elb:listener:8080:
    ListenerProtocol: HTTP
    InstanceProtocol: HTTP
    InstancePort: 8080
```

如果您的环境在您创建的自定义 Amazon Virtual Private Cloud (Amazon VPC) 中运行，则 Elastic Beanstalk 会处理其余任务。在默认 VPC 中，您需要配置实例的安全组以允许来自负载均衡器的入口流量。添加将入口流量规则添加到安全组的第二个配置文件：

.ebextensions/elb-ingress.config

```
Resources:
  port8080SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 8080
      FromPort: 8080
      SourceSecurityGroupName: { "Fn::GetAtt": ["AWSEBLoadBalancer",
 "SourceSecurityGroup.GroupName"] }
```

有关配置文件格式的更多信息，请参阅添加和自定义 Elastic Beanstalk 环境资源和选项设置。

除了将侦听器添加到 Elastic Load Balancing 配置并在安全组中打开端口外，您还需要将主机实例上的端口映射到 Dockerrun.aws.json v2 文件 containerDefinitions 部分的 Docker 容器上的端口。以下摘录显示了一个示例：

```
"portMappings": [
  {
    "hostPort": 8080,
    "containerPort": 8080
  }
]
```

有关 Dockerrun.aws.json v2 文件格式的详细信息，请参阅 Dockerrun.aws.json v2。

失败的容器部署

如果 Amazon ECS 任务失败，您的 Elastic Beanstalk 环境中的一个或多个容器将不会启动。由于 Amazon ECS 任务失败，Elastic Beanstalk 不会回滚多容器环境。如果容器在您的环境中无法启动，请从 Elastic Beanstalk 控制台中重新部署当前版本或以前的正常工作版本。

部署现有版本

1. 在您环境的区域中打开 Elastic Beanstalk 控制台。

2. 单击应用程序名称右侧的操作，然后单击查看应用程序版本。
3. 选择应用程序的版本，然后单击部署。

ECS 托管式 Docker 配置

Dockerrun.aws.json 是 Elastic Beanstalk 配置文件，描述如何将一组托管在 ECS 集群中的 Docker 容器部署到 Elastic Beanstalk 环境中。Elastic Beanstalk 平台创建 ECS 任务定义，其中包括 ECS 容器定义。这些定义在 Dockerrun.aws.json 配置文件中描述。

Dockerrun.aws.json 文件中的容器定义描述了要部署到 ECS 集群中每个 Amazon EC2 实例的容器。在这种情况下，Amazon EC2 实例也被称为主机容器实例，因为其托管 Docker 容器。配置文件还描述了要在主机容器实例上创建的用于挂载 Docker 容器的数据卷。有关 Elastic Beanstalk 上的 ECS 托管 Docker 环境中的组件的更多信息和示意图，请参阅本章前面的 ECS 托管的 Docker 平台。

Dockerrun.aws.json 文件既可单独使用，也可与其他源代码一起压缩到一个存档中。与 Dockerrun.aws.json 一起存档的源代码将部署到 Amazon EC2 容器实例中，并且可在 /var/app/current/ 目录中进行访问。

主题

Dockerrun.aws.json v2

Dockerrun.aws.json 文件包含以下部分：

AWSEBDockerrunVersion

对于 ECS 托管式 Docker 环境，请将版本号指定为值 2。

卷

从 Amazon EC2 容器实例中的文件夹或从源包（部署到 /var/app/current）创建卷。使用 containerDefinitions 部分中的 mountPoints 将这些卷装载到您的 Docker 容器中的路径。

containerDefinitions

容器定义数组。

身份验证（可选）

包含私有存储库身份验证数据的 `.dockercfg` 文件在 Amazon S3 中的位置。

`Dockerrun.aws.json` 的 containerDefinitions 和卷部分使用的格式与 Amazon ECS 任务定义文件的相应部分使用的格式相同。有关任务定义格式和完整任务定义参数列表的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 任务定义。

卷格式

volume 参数从 Amazon EC2 容器实例中的文件夹或从源包（部署到 `/var/app/current`）创建卷。

用以下格式指定卷：

```
"volumes": [
    {
      "name": "volumename",
      "host": {
        "sourcePath": "/path/on/host/instance"
      }
    }
  ],
```

使用容器定义中的 `mountPoints`，将这些卷装载到您的 Docker 容器中的路径。

Elastic Beanstalk 为日志配置附加卷（为每个容器配置一个）。这些卷应由您的 Docker 容器装载，以便将日志写入主机实例。

有关更多详细信息，请参阅后面的容器定义格式部分中的 `mountPoints` 字段。

容器定义格式

以下示例显示 containerDefinitions 部分中常用参数的子集。将提供更多可选参数。

Beanstalk 平台创建 ECS 任务定义，其中包括 ECS 容器定义。Beanstalk 支持 ECS 容器定义的参数子集。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的容器定义。

`Dockerrun.aws.json` 文件包含一组 (一个或多个) 带以下字段的容器定义对象：

name

容器的名称。有关最大长度和允许字符数的信息，请参阅标准容器定义参数。

image

您从中构建 Docker 容器的在线 Docker 存储库中的 Docker 映像的名称。记下这些约定：

- Docker Hub 上的官方存储库中的映像使用一个名称 (例如，ubuntu 或 mongo)。
- Docker Hub 上其他存储库中的映像通过组织名称 (例如，amazon/amazon-ecs-agent) 进行限定。
- 其他在线存储库中的映像由域名 (例如，quay.io/assemblyline/ubuntu) 进行进一步限定。

环境

一组要传递到容器的环境变量。

例如，以下条目定义名称为 **Container** 且值为 **PHP** 的环境变量：

```
"environment": [
  {
    "name": "Container",
    "value": "PHP"
  }
],
```

essential

如果在容器出现故障时应停止任务，则为 true。非主要容器可在不影响实例上的其余容器的情况下完成或发生崩溃。

memory

要为容器预留的容器实例上的内存量。在容器定义中为 memory 和/或 memoryReservation 参数指定一个非零整数。

memoryReservation

要为容器预留的内存量的软限制（以 MiB 为单位）。在容器定义中为 memory 和/或 memoryReservation 参数指定一个非零整数。

mountPoints

Amazon EC2 容器实例中要挂载的卷，以及 Docker 容器文件系统上要挂载这些卷的位置。如果您装载的卷中包含应用程序内容，您的容器能够读取您在源包中上传的数据。当您挂载日志卷以写入日志数据时，Elastic Beanstalk 可以从这些卷中收集日志数据。

Elastic Beanstalk 在 `/var/log/containers/`*`containername`* 处的容器实例上创建日志卷（为每个 Docker 容器创建一个日志卷）。这些卷将命名为 `awseb-logs-`*`containername`* 且将挂载到将日志写入到的容器文件结构中的位置。

例如，以下挂载点将容器中的 nginx 日志位置映射到 Elastic Beanstalk 为 `nginx-proxy` 容器生成的卷。

```
{
  "sourceVolume": "awseb-logs-nginx-proxy",
  "containerPath": "/var/log/nginx"
}
```

portMappings

将容器上的网络端口映射到主机上的端口。

links

要链接到的容器的列表。链接的容器可互相发现并安全地通信。

volumesFrom

从不同容器装载所有卷。例如，从名为 `web` 的容器装载卷：

```
"volumesFrom": [
  {
    "sourceContainer": "web"
  }
],
```

验证库格式 – 使用私有存储库中的映像

`authentication` 部分包含私有存储库的身份验证数据。此条目是可选项。

在 `authentication` 文件的 `Dockerrun.aws.json` 参数中添加有关包含身份验证文件的 Amazon S3 存储桶的信息。确保 `authentication` 参数包含有效的 Amazon S3 存储桶和密钥。Amazon S3 存储桶必须托管在使用它的环境所在的区域中。Elastic Beanstalk 不会从托管在其他区域的 Amazon S3 存储桶下载文件。

使用下列格式：

```
"authentication": {
    "bucket": "DOC-EXAMPLE-BUCKET",
```

```
    "key": "mydockercfg"
  },
```

有关生成和上传身份验证文件的信息，请参阅本章的环境配置主题中的 使用私有存储库中的映像。

示例 Dockerrun.aws.json v2

以下代码段是一个示例，该示例说明了带两个容器实例的 Dockerrun.aws.json 文件的语法。

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/current/proxy/conf.d"
      }
    }
  ],
  "containerDefinitions": [
    {
      "name": "php-app",
      "image": "php:fpm",
      "environment": [
        {
          "name": "Container",
          "value": "PHP"
        }
      ],
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
```

```
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        },
        {
          "sourceVolume": "nginx-proxy-conf",
          "containerPath": "/etc/nginx/conf.d",
          "readOnly": true
        },
        {
          "sourceVolume": "awseb-logs-nginx-proxy",
          "containerPath": "/var/log/nginx"
        }
      ]
    }
  ]
}
```

使用 Elastic Beanstalk 控制台启动 ECS 托管式 Docker 环境

您可以使用 Elastic Beanstalk 控制台在单实例或可扩展的 Elastic Beanstalk 环境中启动多容器实例集群。本教程详细说明了容器配置和使用两个容器的环境的源代码准备。

容器、PHP 应用程序和 nginx 代理在 Elastic Beanstalk 环境中的每个 Amazon Elastic Compute Cloud (Amazon EC2) 实例上并排运行。创建环境并验证应用程序正在运行后，您将连接到一个容器实例以了解其配合使用情况。

小节目录

定义 ECS 托管式 Docker 容器

创建新的 Docker 环境的第一步是为您的应用程序数据创建目录。此文件夹可以位于您的本地计算机上的任何位置，并且可以具有您选择的任何名称。除了容器配置文件外，此文件夹还包含您将上传到 Elastic Beanstalk 并部署到环境的内容。

> **Note**
>
> 此教程的所有代码都可在 GitHub（网址为 https://github.com/awslabs/eb-docker-nginx-proxy）上的 awslab 存储库中找到。

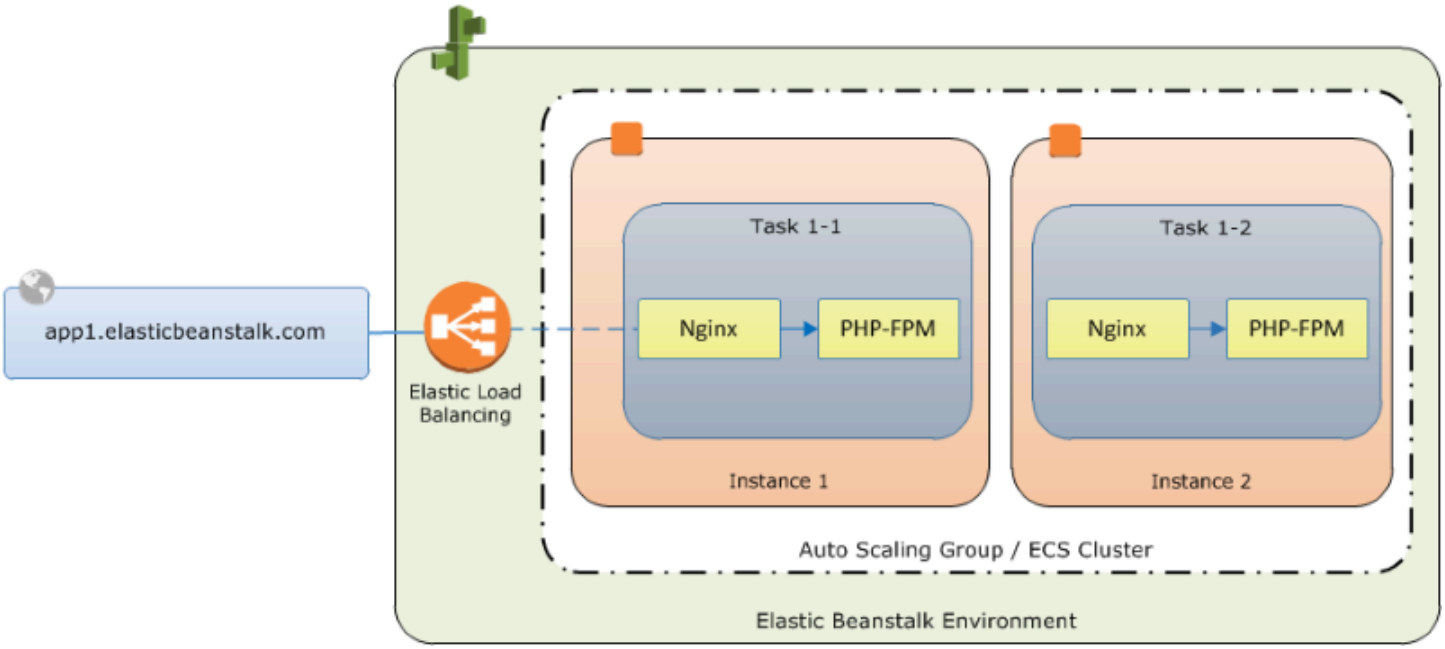
Elastic Beanstalk 用于在 Amazon EC2 实例上配置容器的文件是一个名为 `Dockerrun.aws.json` 的 JSON 格式的文本文件。请使用此名称在应用程序的根目录下创建一个文本文件添加以下文字：

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/current/proxy/conf.d"
      }
    }
  ],
  "containerDefinitions": [
```

```
    {
      "name": "php-app",
      "image": "php:fpm",
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        },
        {
          "sourceVolume": "nginx-proxy-conf",
          "containerPath": "/etc/nginx/conf.d",
          "readOnly": true
        },
        {
          "sourceVolume": "awseb-logs-nginx-proxy",
          "containerPath": "/var/log/nginx"
        }
      ]
    }
```

```
  ]
}
```

本示例配置定义了两个容器：一个 PHP 网站以及一个位于它前面的 nginx 代理。这两个容器将在 Elastic Beanstalk 环境中每个实例上的 Docker 容器中并排运行，并从主机实例上的卷（也在此文件中定义）访问共享内容（网站的内容）。这两个容器自身从托管于 Docker 中心上的正式存储库中的映像创建。得到的环境与下面类似：



在配置中定义的卷与您接下来要创建并作为应用程序源包的一部分上传的内容相对应。这两个容器将通过在容器定义的 `mountPoints` 部分中装载卷来访问主机上的内容。

有关 `Dockerrun.aws.json` 的格式及其参数的更多信息，请参阅。[容器定义格式]

添加内容

接下来，您将为要向访问者显示的 PHP 站点添加一些内容，并为 nginx 代理添加一个配置文件。

php-app/index.php

```
<h1>Hello World!!!</h1>
<h3>PHP Version <pre><?= phpversion()?></pre></h3>
```

php-app/static.html

```
<h1>Hello World!</h1>
```

```
<h3>This is a static HTML page.</h3>
```

proxy/conf.d/default.conf

```
server {
  listen 80;
  server_name localhost;
  root /var/www/html;

  index index.php;

  location ~ [^/]\.php(/|$) {
    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
    if (!-f $document_root$fastcgi_script_name) {
      return 404;
    }

    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_param PATH_TRANSLATED $document_root$fastcgi_path_info;

    fastcgi_pass php-app:9000;
    fastcgi_index index.php;
  }
}
```

部署到 Elastic Beanstalk

您的应用程序文件夹现在包含以下文件：

```
### Dockerrun.aws.json
### php-app
#   ### index.php
#   ### static.html
### proxy
    ### conf.d
        ### default.conf
```

这就是您创建 Elastic Beanstalk 环境所需的全部内容。创建上述文件和文件夹 (不包括顶级项目文件夹) 的 `.zip` 存档。要在 Windows 资源管理器中创建存档，请选择项目文件夹的内容，右键单击，选择 Send To (发送到)，然后单击 Compressed (zipped) Folder (压缩的文件夹)

> **Note**
>
> 有关在其他环境中创建存档所需的文件结构和说明的信息，请参阅创建应用程序源包。

接下来，将源包上传到 Elastic Beanstalk 并创建您的环境。对于 Platform (平台)，选择 Docker。对于平台分支，选择在 64 位 Amazon Linux 2 上运行的 ECS。

启动环境 (控制台)

1. 使用下面的预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced
2. 对于平台，请选择与应用程序使用的语言匹配的平台和平台分支，或者为基于容器的应用程序选择 Docker 平台。
3. 对于 Application code（应用程序代码），选择 Upload your code（上传代码）。
4. 选择 Local file (本地文件)，再选择 Choose file (选择文件)，然后打开源包。
5. 选择复查并启动。
6. 查看可用设置并选择 Create app (创建应用程序)。

Elastic Beanstalk 控制台将您重定向到新环境的管理控制面板。此屏幕显示环境的运行状况和由 Elastic Beanstalk 服务输出的事件。当状态为 Green 时，请单击环境名称旁边的 URL 以查看您新的网站。

连接到容器实例

接下来，您将连接到 Elastic Beanstalk 环境中的 Amazon EC2 实例，以查看某些移动部件的实际应用。

要连接到您的环境中的实例，最简单的方法是使用 EB CLI。要使用该工具，请安装 EB CLI（如果尚未安装）。还需要为您的环境配置一个 Amazon EC2 SSH 密钥对。可以使用控制台的安全配置页面或 EB CLI eb init 命令执行该操作。要连接到环境实例，请使用 EB CLI eb ssh 命令。

现在，您已连接到托管 Docker 容器的 Amazon EC2 实例，您可以查看设置情况。对 `ls` 运行 `/var/app/current`：

```
[ec2-user@ip-10-0-0-117 ~]$ ls /var/app/current
Dockerrun.aws.json  php-app  proxy
```

此目录包含您在创建环境期间上传到 Elastic Beanstalk 的源包中的文件。

```
[ec2-user@ip-10-0-0-117 ~]$ ls /var/log/containers
nginx-proxy    nginx-proxy-4ba868dbb7f3-stdouterr.log
php-app        php-app-dcc3b3c8522c-stdouterr.log       rotated
```

这是在容器实例上创建日志并由 Elastic Beanstalk 收集日志的位置。Elastic Beanstalk 在此目录中为每个容器创建一个卷，您将卷挂载到写入日志的容器位置。

您还可以使用 `docker ps` 查看 Docker 以了解正在运行的容器。

```
[ec2-user@ip-10-0-0-117 ~]$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
  STATUS                  PORTS                                  NAMES

4ba868dbb7f3   nginx                                "/docker-entrypoint.…"   4 minutes ago
   Up 4 minutes            0.0.0.0:80->80/tcp, :::80->80/tcp   ecs-awseb-Tutorials-env-
dc2aywfjwg-1-nginx-proxy-acca84ef87c4aca15400
dcc3b3c8522c   php:fpm                              "docker-php-entrypoi…"   4 minutes ago
   Up 4 minutes            9000/tcp                               ecs-awseb-Tutorials-env-
dc2aywfjwg-1-php-app-b8d38ae288b7b09e8101
d9367c0baad6   amazon/amazon-ecs-agent:latest   "/agent"                 5 minutes ago
  Up 5 minutes (healthy)                                          ecs-agent
```

这会显示您部署的两个正在运行的容器，以及协调部署的 Amazon ECS 容器代理。

检查 Amazon ECS 容器代理

Elastic Beanstalk 上的 ECS 托管式 Docker 环境中的 Amazon EC2 实例在 Docker 容器中运行代理进程。此代理连接到 Amazon ECS 服务以协调容器部署。这些部署在 Amazon ECS 中作为任务运行，这些任务在任务定义文件中进行配置。Elastic Beanstalk 根据您在源包中上传的 `Dockerrun.aws.json` 创建这些任务定义文件。

利用对 `http://localhost:51678/v1/metadata` 的 HTTP 获取请求查看容器代理的状态：

```
[ec2-user@ip-10-0-0-117 ~]$ curl http://localhost:51678/v1/metadata
{
  "Cluster":"awseb-Tutorials-env-dc2aywfjwg",
  "ContainerInstanceArn":"arn:aws:ecs:us-west-2:123456789012:container-instance/awseb-
Tutorials-env-dc2aywfjwg/db7be5215cd74658aacfcb292a6b944f",
  "Version":"Amazon ECS Agent - v1.57.1 (089b7b64)"
}
```

此结构显示 Amazon ECS 集群的名称以及集群实例（您连接到的 Amazon EC2 实例）的 ARN（Amazon 资源名称）。

有关更多信息，请发出 HTTP 获取请求，地址为 `http://localhost:51678/v1/tasks`：

```
[ec2-user@ip-10-0-0-117 ~]$ curl http://localhost:51678/v1/tasks
{
   "Tasks":[
      {
         "Arn":"arn:aws:ecs:us-west-2:123456789012:task/awseb-Tutorials-env-dc2aywfjwg/
bbde7ebe1d4e4537ab1336340150a6d6",
         "DesiredStatus":"RUNNING",
         "KnownStatus":"RUNNING",
         "Family":"awseb-Tutorials-env-dc2aywfjwg",
         "Version":"1",
         "Containers":[
            {

 "DockerId":"dcc3b3c8522cb9510b7359689163814c0f1453b36b237204a3fd7a0b445d2ea6",
               "DockerName":"ecs-awseb-Tutorials-env-dc2aywfjwg-1-php-app-
b8d38ae288b7b09e8101",
               "Name":"php-app",
               "Volumes":[
                  {
                     "Source":"/var/app/current/php-app",
                     "Destination":"/var/www/html"
                  }
               ]
            },
            {

 "DockerId":"4ba868dbb7f3fb3328b8afeb2cb6cf03e3cb1cdd5b109e470f767d50b2c3e303",
               "DockerName":"ecs-awseb-Tutorials-env-dc2aywfjwg-1-nginx-proxy-
acca84ef87c4aca15400",
               "Name":"nginx-proxy",
               "Ports":[
                  {
                     "ContainerPort":80,
                     "Protocol":"tcp",
                     "HostPort":80
                  },
                  {
                     "ContainerPort":80,
                     "Protocol":"tcp",
```

```
                    "HostPort":80
                }
            ],
            "Volumes":[
                {
                    "Source":"/var/app/current/php-app",
                    "Destination":"/var/www/html"
                },
                {
                    "Source":"/var/log/containers/nginx-proxy",
                    "Destination":"/var/log/nginx"
                },
                {
                    "Source":"/var/app/current/proxy/conf.d",
                    "Destination":"/etc/nginx/conf.d"
                }
            ]
        }
    ]
  }
]
}
```

此结构描述了为部署本教程的示例项目中的两个 docker 容器而运行的任务。将显示以下信息：

- KnownStatus – RUNNING 状态表示容器仍处于活动状态。
- Family – Elastic Beanstalk 从 Dockerrun.aws.json 中创建的任务定义的名称。
- Version – 任务定义的版本。每当任务定义文件更新时，版本号都会递增。
- Containers – 有关在实例上运行的容器的信息。

甚至可以从 Amazon ECS 服务本身获取更多信息，您可以使用 调用该服务AWS Command Line Interface 有关将 AWS CLI 与 Amazon ECS 结合使用的说明以及有关 Amazon ECS 的一般信息，请参阅 Amazon ECS 用户指南。

将在 Amazon Linux 上运行的多容器 Docker 迁移到 Amazon Linux 2023 上的 ECS

2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。这包括平台分支 64 位 Amazon Linux 上运行的多容器 Docker。本主题将指导您将应用程序从此已停用平台分支迁移到在 64 位 AL2023 上运行的 ECS。此目标平台分支是最新的并且受支持。

类似于先前的多容器 Docker AL1 分支，新的 ECS AL2023 平台分支使用 Amazon ECS，将多个 Docker 容器协调部署到 Elastic Beanstalk 环境中的 Amazon ECS 集群。新的 ECS AL2023 平台分支支持先前的多容器 Docker AL1 平台分支的所有功能。此外，还支持相同的 `Dockerrun.aws.json` v2 文件。

小节目录

使用 Elastic Beanstalk 控制台来迁移

要使用 Elastic Beanstalk 控制台进行迁移，请将相同的源代码部署到基于在 AL2023 上运行的 ECS 平台分支的新环境。源代码无需更改。

迁移到在 Amazon Linux 2023 上运行的 ECS 平台分支

1. 使用已部署到旧环境的应用程序源，创建应用程序源包。您可以使用相同的应用程序源包和相同的 `Dockerrun.aws.json` v2 文件。
2. 使用在 Amazon Linux 2023 上运行的 ECS 平台分支创建新环境。将上一步骤中的源捆绑包用于应用程序代码。有关更多详细步骤，请参阅本章前面 ECS 托管的 Docker 教程中的 部署到 Elastic Beanstalk。

使用 AWS CLI 迁移

您还可以选择使用 AWS Command Line Interface（AWS CLI）将现有的多容器 Docker Amazon Linux Docker 环境迁移到更新的 ECS AL2023 平台分支。在这种情况下，您无需创建新环境或重新部署源代码。您只需运行 AWS CLI update-environment 命令。它将执行平台更新，以将现有环境迁移到 ECS Amazon Linux 2023 平台分支。

使用以下语法将环境迁移到新的平台分支。

```
aws elasticbeanstalk update-environment \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2023 version running ECS" \
--region my-region
```

以下命令示例是在 us-east-1 区域将环境 beta-101 迁移到 版本 3.0.0 的 ECS Amazon Linux 2023 平台分支。

```
aws elasticbeanstalk update-environment \
--environment-name beta-101 \
--solution-stack-name "64bit Amazon Linux 2023 v4.0.0 running ECS" \
--region us-east-1
```

solution-stack-name 参数提供平台分支及其版本。通过指定合适的解决方案堆栈名称来使用最新的平台分支版本。每个平台分支的版本都包含在解决方案堆栈名称中，如上例所示。有关 Docker 平台最新解决方案堆栈的列表，请参阅AWS Elastic Beanstalk平台指南中的支持平台。

> **Note**
>
> list-available-solution-stacks 命令提供 AWS 区域中您的账户可用的平台版本列表。
>
> ```
> aws elasticbeanstalk list-available-solution-stacks --region us-east-1 --query
>  SolutionStacks
> ```

要了解有关 AWS CLI 的更多信息，请参阅AWS Command Line Interface用户指南。有关 Elastic Beanstalk AWS CLI 命令的更多信息，请参阅 AWS CLIElastic Beanstalk 命令参考。

（旧式）从在 Amazon Linux 上运行的多容器 Docker 迁移到在 Amazon Linux 2 上运行的 Docker 平台分支

在 64 位 Amazon Linux 2 上运行的 ECS 平台分支发布之前，针对拥有基于在 64 位 Amazon Linux 上运行的多容器 Docker 平台分支的环境的客户，Elastic Beanstalk 提供了到 Amazon Linux 2 的备用迁移路径。本主题介绍了该迁移路径，并保留在本文档中，作为完成该迁移路径的任何客户的参考。

如果客户拥有基于在 64 位 Amazon Linux 上运行的多容器 Docker 平台分支的环境，我们现在建议迁移到在 64 位 Amazon Linux 2 上运行 ECS 平台分支。与备用迁移路径不同，此方法继续使用 Amazon ECS 来协调向 ECS 托管式 Docker 环境的容器部署。这方面允许采用更直接的方法。无需更改源代码，支持相同的 Dockerrun.aws.json v2。有关更多信息，请参阅 将在 Amazon Linux 上运行的多容器 Docker 迁移到 Amazon Linux 2023 上的 ECS。

从 Amazon Linux 上的多容器 Docker 到 Docker Amazon Linux 2 平台分支的旧式迁移

您可以将在 Amazon Linux AMI 上的多容器 Docker 平台上运行的应用程序迁移到 Amazon Linux 2 Docker 平台。Amazon Linux AMI 上的多容器 Docker 平台要求您指定要作为容器运行的预构建应用程序映像。迁移后，您将不再有此限制，因为 Amazon Linux 2 Docker 平台还允许 Elastic Beanstalk 在

部署期间构建容器映像。您的应用程序将继续在多容器环境中运行，并获得 Docker Compose 工具的额外优势。

Docker Compose 是定义和运行多容器 Docker 应用程序的工具。要了解有关 Docker Compose 以及如何安装它的更多信息，请参阅 Docker 网站 Docker Compose 概述和安装 Docker Compose。

`docker-compose.yml` 文件

Docker Compose 工具使用 `docker-compose.yml` 文件配置您的应用程序服务。此文件将替换应用程序项目目录和应用程序源包中的 `Dockerrun.aws.json v2` 文件。您可以手动创建 `docker-compose.yml` 文件，此时您将发现这对引用您的 `Dockerrun.aws.json v2` 文件以获取大多数参数值很有帮助。

以下是同一应用程序的 `docker-compose.yml` 文件和相应 `Dockerrun.aws.json v2` 文件的示例。有关 `docker-compose.yml` 文件的更多信息，请参阅 Compose 文件参考。有关 `Dockerrun.aws.json v2` 文件的更多信息，请参阅Dockerrun.aws.json v2。

`docker-compose.yml`

```
version: '2.4'
services:
  php-app:
    image: "php:fpm"
    volumes:
      - "./php-app:/var/www/html:ro
"
      - "${EB_LOG_BASE_DIR}/php-app
:/var/log/sample-app"
    mem_limit: 128m
    environment:
      Container: PHP
  nginx-proxy:
    image: "nginx"
    ports:
      - "80:80"
    volumes:
      - "./php-app:/var/www/html:ro
"
```

`Dockerrun.aws.json v2`

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/
current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/
current/proxy/conf.d"
      }
    }
  ],
  "containerDefinitions": [
    {
```

docker-compose.yml

```
      - "./proxy/conf.d:/etc/nginx/
conf.d:ro"
      - "${EB_LOG_BASE_DIR}/nginx-p
roxy:/var/log/nginx"
    mem_limit: 128m
    links:
      - php-app
```

Dockerrun.aws.json v2

```
      "name": "php-app",
      "image": "php:fpm",
      "environment": [
        {
          "name": "Container",
          "value": "PHP"
        }
      ],
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app"
,
          "containerPath": "/var/www
/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        {
          "sourceVolume": "php-app"
,
          "containerPath": "/var/www
/html",
          "readOnly": true
        },
```

docker-compose.yml	Dockerrun.aws.json v2

```
        {
          "sourceVolume": "nginx-pr
oxy-conf",
          "containerPath": "/etc/ngi
nx/conf.d",
          "readOnly": true
        },
        {
          "sourceVolume": "awseb-lo
gs-nginx-proxy",
          "containerPath": "/var/log
/nginx"
        }
      ]
    }
  ]
}
```

其他迁移注意事项

Docker Amazon Linux 2 平台和多容器 Docker Amazon Linux AMI 平台以不同方式实现环境属性。这两个平台还有 Elastic Beanstalk 为其每个容器创建的不同日志目录。从 Amazon Linux AMI 多容器 Docker 平台迁移后，您需要了解这些不同的实现方式，以便在新 Amazon Linux 2 Docker 平台环境中使用。

区域图	Amazon Linux 2 上带 Docker Compose 的 Docker 平台	Amazon Linux AMI 上的多容器 Docker 平台
环境属性	为使容器能够访问环境属性，您必须在 `.env` 文件中添加对 `docker-compose.yml` 文件的引用。Elastic Beanstalk 生成 `.env` 文件，将每个属性列为环境变量。有关更多信息，请参阅在容器中引用环境变量。	Elastic Beanstalk 可以直接将环境属性传递给容器。在容器中运行的代码可以将这些属性作为环境变量访问，而无需任何其他配置。

区域图	Amazon Linux 2 上带 Docker Compose 的 Docker 平台	Amazon Linux AMI 上的多容器 Docker 平台
日志目录	对于每个容器，Elastic Beanstalk 都会创建一个名为 /var/log/eb-docker/containers/ *<service name>* (或 ${EB_LOG_BASE_DIR}/<service name>) 的日志目录。有关更多信息，请参阅Docker 容器自定义日志记录 (Docker Compose)。	对于每个容器，Elastic Beanstalk 都会创建一个名为 /var/log/containers/ *<containername>* 的日志目录。有关更多信息，请参阅容器定义格式中的 mountPoints 字段。

迁移步骤

迁移到 Amazon Linux 2 Docker 平台

1. 根据应用程序的现有 docker-compose.yml 文件为应用程序创建 Dockerrun.aws.json v2 文件。有关更多信息，请参阅上述 docker-compose.yml 文件部分。
2. 在应用程序项目文件夹的根目录中，将 Dockerrun.aws.json v2 文件替换为刚刚创建的 docker-compose.yml 文件。

   您的目录结构应如下所示。

   ```
   ~/myApplication
   |-- docker-compose.yml
   |-- .ebextensions
   |-- php-app
   |-- proxy
   ```

3. 使用 eb init 命令配置本地目录以部署到 Elastic Beanstalk。

   ```
   ~/myApplication$ eb init -p docker application-name
   ```

4. 使用 eb create 命令创建环境并部署 Docker 映像。

   ```
   ~/myApplication$ eb create environment-name
   ```

5. 如果您的应用是 Web 应用程序，则在启动环境后，使用 eb open 命令在 Web 浏览器中查看它。

```
~/myApplication$ eb open environment-name
```

6. 您可以使用 eb status 命令显示新创建的环境的状态。

```
~/myApplication$ eb status environment-name
```

预配置的 Docker 容器 (Amazon Linux AMI)

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

不再支持在亚马逊 Linux AMI (AL1) 上运行的预配置 Docker GlassFish 平台分支。要将您的 GlassFish 应用程序迁移到支持的亚马逊 Linux 2023 平台，请将您的应用程序代码部署 GlassFish 到亚马逊 Linux 2023 Docker 镜像。有关更多信息，请参阅以下主题：the section called “教程—— GlassFish 在 Docker 上：2023 年亚马逊 Linux 之路”。

开始使用预配置 Docker 容器 - Amazon Linux AMI（Amazon Linux 2 之前）上

本部分介绍如何在本地开发示例应用程序，然后使用预配置的 Docker 容器将应用程序部署到 Elastic Beanstalk。

设置本地开发环境

在本演练中，我们使用了一个 GlassFish 示例应用程序。

设置环境

1. 为示例应用程序创建新文件夹。

```
~$ mkdir eb-preconf-example
~$ cd eb-preconf-example
```

2. 将示例应用程序代码下载到新文件夹。

```
~$ wget https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/samples/docker-glassfish-v1.zip
~$ unzip docker-glassfish-v1.zip
~$ rm docker-glassfish-v1.zip
```

在本地开发和测试

开发示例 GlassFish 应用程序

1. 将 Dockerfile 添加到应用程序的根目录文件夹。在文件中，指定用于运行本地预配置的 AWS Elastic Beanstalk Docker 容器的 Docker 基础镜像。稍后你将把你的应用程序部署到 Elastic Beanstal GlassFish k 预配置的 Docker 平台版本。选择此平台版本使用的 Docker 基本映像。要了解此平台版本的当前 Docker 镜像的更多信息，请参阅 AWS Elastic Beanstalk 平台指南中的 AWS Elastic Beanstalk 支持的平台页面的预配置 Docker 部分。

   Example ~/E/b-preconf-exampleDockerfile

   ```
   # For Glassfish 5.0 Java 8
   FROM amazon/aws-eb-glassfish:5.0-al-onbuild-2.11.1
   ```

   有关使用 Dockerfile 的更多信息，请参阅Docker 配置。

2. 构建 Docker 映像。

   ```
   ~/eb-preconf-example$ docker build -t my-app-image .
   ```

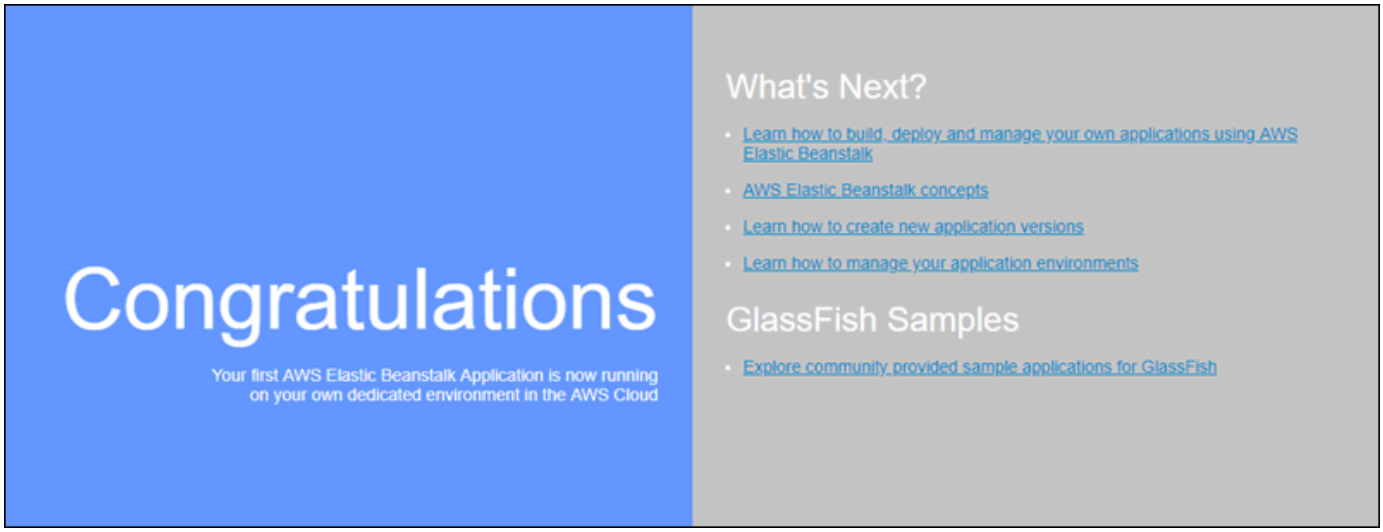
3. 从该映像运行 Docker 容器。

   > **Note**
   >
   > 必须包含 -p 标记才能将容器中的端口 8080 映射到 localhost 端口 3000。Elastic Beanstalk Docker 容器始终在容器上的端口 8080 上公开应用程序。-it 标志将映像作为交互式进程运行。当容器存在时，--rm 标志将清理容器文件系统。您可以选择包含 -d 标志以将映像作为守护程序运行。

   ```
   $ docker run -it --rm -p 3000:8080 my-app-image
   ```

4. 要查看示例应用程序，请将以下 URL 键入您的 Web 浏览器。

```
http://localhost:3000
```



部署到 Elastic Beanstalk

在测试应用程序后，可随时将其部署到 Elastic Beanstalk。

将应用程序部署到 Elastic Beanstalk

1. 在应用程序的根文件夹中，将 `Dockerfile` 重命名为 `Dockerfile.local`。Elastic Beanstalk 若要使用 `Dockerfile`，则必须执行此步骤，该文件中包含关于 Elastic Beanstalk 在 Elastic Beanstalk 环境中的每个 Amazon EC2 实例上构建自定义 Docker 映像的正确说明。

> **Note**
>
> 如果您的 `Dockerfile` 包含修改平台版本的基础 Docker 映像的指令，则无需执行此步骤。如果您的 `Dockerfile` 仅包含用来指定从中构建容器的基本映像的 `Dockerfile` 行，则完全无需使用 `FROM`。在这种情况下，`Dockerfile` 是冗余的。

2. 创建应用程序源包。

```
~/eb-preconf-example$ zip myapp.zip -r *
```

3. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &Environmentype= LoadBalanced

4. 对于 Platform (平台)，在 Preconfigured – Docker (预配置 – Docker) 下，选择 Glassfish。
5. 对于 Application code (应用程序节点)，选择 Upload your code (上传您的节点)，然后选择 Upload (上传)。
6. 选择 Local file (本地文件)，再选择 Browse (浏览)，然后打开您刚刚创建的应用程序源包。
7. 选择上传。
8. 选择复查并启动。
9. 查看可用设置并选择 Create app (创建应用程序)。
10. 创建环境后，您可以查看已部署的应用程序。选择控制台控制面板顶部显示的环境 URL。

将 GlassFish 应用程序部署到 Docker 平台：2023 年亚马逊 Linux 的迁移之路

本教程的目标是为使用预配置的 Docker GlassFish 平台（基于亚马逊 Linux AMI）的客户提供向亚马逊 Linux 2023 的迁移路径。您可以通过部署 GlassFish 将 GlassFish 应用程序迁移到亚马逊 Linux 2023，将应用程序代码迁移到亚马逊 Linux 2023 Docker 镜像。

本教程将引导你使用 AWS Elastic Beanstalk Docker 平台将基于 Java EE 应用服务器的应用程序部署到 E GlassFish lastic Beanstalk 环境中。

我们演示两种构建 Docker 映像的方法：

- 简单 — 提供您的 GlassFish 应用程序源代码，让 Elastic Beanstalk 在配置环境的过程中构建和运行 Docker 映像。这很容易设置，但代价是增加实例预置时间。
- 高级 – 构建包含应用程序代码和依赖关系的自定义 Docker 映像，并将其提供给 Elastic Beanstalk 以在您的环境中使用。这种方法的参与度稍高一些，并减少了环境中实例的预置时间。

先决条件

本教程假设您对基本 Elastic Beanstalk 操作、Elastic Beanstalk 命令行界面（EB CLI）和 Docker 有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。本教程使用 EB CLI，但您也可以使用 Elastic Beanstalk 控制台创建环境并上传应用程序。

要完成本教程，您需要做以下 Docker 组件：

- 在本地安装有效的 Docker。有关更多信息，请参阅 Docker 文档网站上的获取 Docker。
- 访问 Docker Hub。您需要创建一个 Docker ID 才能访问 Docker Hub。有关更多信息，请参阅 Docker 文档网站上的共享应用程序。

要了解有关在 Elastic Beanstalk 平台上配置 Docker 环境的更多信息，请参阅同一章中的Docker 配置。

简单示例：提供您的应用程序代码

这是部署 GlassFish 应用程序的简便方法。提供您的应用程序源代码以及本教程中包含的 `Dockerfile`。Elastic Beanstalk 会生成一个 Docker 镜像，其中包含你的应用程序和软件堆栈。GlassFish 然后，Elastic Beanstalk 在环境实例上运行该映像。

此方法的一个问题是，无论 Elastic Beanstalk 何时为您的环境创建实例，都会在本地构建 Docker 映像。映像构建会增加实例预置时间。这种影响不限于初始环境创建，也会在扩展操作期间发生。

使用示例 GlassFish 应用程序启动环境

1. 下载示例 `docker-glassfish-al2-v1.zip`，然后将 `.zip` 文件展开到开发环境的目录中。

```
~$ curl https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/samples/docker-
glassfish-al2-v1.zip --output docker-glassfish-al2-v1.zip
~$ mkdir glassfish-example
~$ cd glassfish-example
~/glassfish-example$ unzip ../docker-glassfish-al2-v1.zip
```

   您的目录结构应如下所示。

```
~/glassfish-example
|-- Dockerfile
|-- Dockerrun.aws.json
|-- glassfish-start.sh
|-- index.jsp
|-- META-INF
|   |-- LICENSE.txt
|   |-- MANIFEST.MF
|   `-- NOTICE.txt
|-- robots.txt
`-- WEB-INF
    `-- web.xml
```

   以下文件是在环境中构建和运行 Docker 容器的关键：

   - `Dockerfile` – 提供 Docker 用于构建具有应用程序和所需依赖关系的映像的说明。
   - `glassfish-start.sh` – Docker 映像运行以启动应用程序的 Shell 脚本。

- `Dockerrun.aws.json`— 提供日志密钥，以包括 GlassFish 应用程序服务器登录日志文件请求。如果您对 GlassFish 日志不感兴趣，可以省略此文件。

2. 为到 Elastic Beanstalk 的部署配置本地目录。

```
~/glassfish-example$ eb init -p docker glassfish-example
```

3. （可选）使用 eb local run 命令在本地生成并运行您的容器。

```
~/glassfish-example$ eb local run --port 8080
```

> **Note**
>
> 要了解有关 eb local 命令的更多信息，请参阅 the section called “eb local”。Windows 中不支持此命令。或者，您也可以使用 docker build 和 docker run 命令生成并运行您的容器。有关更多信息，请参阅 Docker 文档。

4. （可选）当您的容器正在运行时，可使用 eb local open 命令在 Web 浏览器中查看您的应用程序。或者，在 Web 浏览器中打开 http://localhost:8080/。

```
~/glassfish-example$ eb local open
```

5. 使用 eb create 命令创建环境并部署应用程序。

```
~/glassfish-example$ eb create glassfish-example-env
```

6. 启动环境后，使用 eb open 命令在 Web 浏览器中查看它。

```
~/glassfish-example$ eb open
```

完成使用该示例后，终止环境并删除相关资源。

```
~/glassfish-example$ eb terminate --all
```

高级示例：提供预构建的 Docker 映像

这是一种更高级的 GlassFish 应用程序部署方式。在第一个示例的基础上，您创建了一个包含应用程序代码和 GlassFish 软件堆栈的 Docker 镜像，然后将其推送到 Docker Hub。完成此一次性步骤后，您可以根据自定义映像启动 Elastic Beanstalk 环境。

当您启动环境并提供 Docker 映像时，环境中的实例会直接下载并使用此映像，而不需要构建 Docker 映像。因此，实例预置时间会减少。

> **注意**
>
> - 以下步骤将创建一个公开可用的 Docker 映像。
> - 您将使用本地安装的 Docker 中的 Docker 命令以及 Docker Hub 凭据。有关更多信息，请参阅本主题前面的 “先决条件” 部分。

使用预构建的 GlassFish 应用程序 Docker 镜像启动环境

1. 下载并展开示例 `docker-glassfish-al2-v1.zip`，如前面的简单示例所示。如果您已完成该示例，则可以使用已有的目录。
2. 构建一个 Docker 映像并将其推送到 Docker Hub。为 *`docker-id`* 输入您的 Docker ID，以登录到 Docker Hub。

   ```
   ~/glassfish-example$ docker build -t docker-id/beanstalk-glassfish-example:latest .
   ~/glassfish-example$ docker push docker-id/beanstalk-glassfish-example:latest
   ```

   > **Note**
   >
   > 在推送映像之前，您可能需要运行 docker login。如果您运行不带参数的命令，系统将提示您输入 Docker Hub 凭据。

3. 创建其他目录。

   ```
   ~$ mkdir glassfish-prebuilt
   ~$ cd glassfish-prebuilt
   ```

4. 将以下示例复制到名为 `Dockerrun.aws.json` 的文件中。

   Example **`~/glassfish-prebuilt/Dockerrun.aws.json`**

   ```
   {
     "AWSEBDockerrunVersion": "1",
     "Image": {
       "Name": "docker-username/beanstalk-glassfish-example"
     },
   ```

```
  "Ports": [
    {
      "ContainerPort": 8080,
      "HostPort": 8080
    }
  ],
  "Logging": "/usr/local/glassfish5/glassfish/domains/domain1/logs"
}
```

5. 为到 Elastic Beanstalk 的部署配置本地目录。

```
~/glassfish-prebuilt$ eb init -p docker glassfish-prebuilt$
```

6. （可选）使用 eb local run 命令在本地运行容器。

```
~/glassfish-prebuilt$ eb local run --port 8080
```

7. （可选）当您的容器正在运行时，可使用 eb local open 命令在 Web 浏览器中查看您的应用程序。或者，在 Web 浏览器中打开 http://localhost:8080/。

```
~/glassfish-prebuilt$ eb local open
```

8. 使用 eb create 命令创建环境并部署 Docker 映像。

```
~/glassfish-prebuilt$ eb create glassfish-prebuilt-env
```

9. 启动环境后，使用 eb open 命令在 Web 浏览器中查看它。

```
~/glassfish-prebuilt$ eb open
```

完成使用该示例后，终止环境并删除相关资源。

```
~/glassfish-prebuilt$ eb terminate --all
```

在 Elastic Beanstalk 上创建和部署 Go 应用程序

AWS Elastic Beanstalk for Go 让您可以使用 Amazon Web Services 轻松部署、管理和扩展您的 Go Web 应用程序。使用 Go 开发或托管 Web 应用程序的任何人都可以使用适用于 Go 的 Elastic Beanstalk。本章介绍 step-by-step 如何将您的 Web 应用程序部署到 Elastic Beanstalk。

部署 Elastic Beanstalk 应用程序后，您可以继续使用 EB CLI 管理应用程序和环境，也可以使用 Elastic Beanstalk 控制台、 AWS CLI或 API。

主题

- QuickStart: 在 Elastic Beanstalk 上部署 Go 应用程序
- 设置 Go 开发环境
- 使用 Elastic Beanstalk Go 平台

QuickStart: 在 Elastic Beanstalk 上部署 Go 应用程序

本 QuickStart 教程将引导你完成创建 Go 应用程序并将其部署到 AWS Elastic Beanstalk 环境的过程。

> **Note**
>
> 本 QuickStart 教程仅用于演示目的。请勿将本教程中创建的应用程序用于生产流量。

Sections

- 你的 AWS 账户
- 先决条件
- 步骤 1：创建 Go 应用程序
- 步骤 2：使用 EB CLI 部署您的 Go 应用程序
- 第 3 步：在 Elastic Beanstalk 上运行你的应用程序
- 步骤 4：清除
- AWS 您的应用程序的资源
- 后续步骤
- 使用 Elastic Beanstalk 控制台进行部署

你的 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

如果您已经有一个 AWS 帐户，则可以继续前进先决条件 。

创建一个 AWS 账户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

先决条件

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

EB CLI

本教程使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

步骤 1：创建 Go 应用程序

创建项目目录。

```
~$ mkdir eb-go
~$ cd eb-go
```

接下来，创建一个您将使用 Elastic Beanstalk 部署的应用程序。我们会创建一个“Hello World”RESTful Web 服务。

本示例输出根据访问服务所使用的路径而变化的自定义问候语。

在此目录中创建名为 `application.go` 的文本文件，包含以下内容：

Example **`~/eb-go/application.go`**

```
package main

import (
 "fmt"
 "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
 if r.URL.Path == "/" {
  fmt.Fprintf(w, "Hello World! Append a name to the URL to say hello. For example, use
 %s/Mary to say hello to Mary.", r.Host)
 } else {
  fmt.Fprintf(w, "Hello, %s!", r.URL.Path[1:])
 }
}

func main() {
 http.HandleFunc("/", handler)
 http.ListenAndServe(":5000", nil)
}
```

步骤 2：使用 EB CLI 部署您的 Go 应用程序

下一步，您将创建应用程序环境并使用 Elastic Beanstalk 部署已配置的应用程序。

创建环境并部署 Go 应用程序

1. 使用 eb init 命令，初始化 EB CLI 存储库。

```
~/eb-go$ eb init -p go go-tutorial --region us-east-2
Application go-tutorial has been created.
```

此命令创建一个名为的应用程序，go-tutorial并将您的本地存储库配置为使用最新 Go 平台版本创建环境。

2. （可选）再次运行 eb init 以配置默认密钥对，以便使用 SSH 连接到运行您的应用程序的 EC2 实例。

```
~/eb-go$ eb init
Do you want to set up SSH for your instances?
(y/n): y
Select a keypair.
1) my-keypair
2) [ Create new KeyPair ]
```

如果您已有密钥对，请选择一个，或按提示创建一个。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。

3. 创建环境并使用 eb create 将应用程序部署到此环境中。Elastic Beanstalk 会自动为您的应用程序生成一个 zip 文件，然后在端口 5000 上启动该文件。

```
~/eb-go$ eb create go-env
```

Elastic Beanstalk 创建您的环境大约需要五分钟。

第 3 步：在 Elastic Beanstalk 上运行你的应用程序

创建环境的过程完成后，使用打开您的网站eb open。

```
~/eb-go$ eb open
```

恭喜您！你已经使用 Elastic Beanstalk 部署了 Go 应用程序！这将使用为应用程序创建的域名打开一个浏览器窗口。

步骤 4：清除

完成应用程序的使用后，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源。

要使用 EB CLI 终止您的 Elastic Beanstalk 环境，请运行以下命令。

```
~/eb-go$ eb terminate
```

AWS 您的应用程序的资源

您刚刚创建了一个单实例应用程序。它可用作带有单个 EC2 实例的简单示例应用程序，因此不需要负载平衡或 auto Scaling。对于单实例应用程序，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

  各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 管理所有这些资源。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。您还可以使用 Elastic Beanstalk 控制台探索您的新环境。有关详细步骤，请参阅本指南“入门”一章中的“探索您的环境”。

部署一到两个示例应用程序并准备好开始在本地开发和运行 Go 应用程序后，请参阅设置 Go 开发环境。

使用 Elastic Beanstalk 控制台进行部署

您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。

设置 Go 开发环境

设置 Go 开发环境以在本地测试应用程序，然后再将它部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

有关适用于所有语言的常见设置步骤和工具，请参阅配置用于 Elastic Beanstalk 的开发计算机。

安装 Go

要在本地运行 Go 应用程序，请装 Go。如果您不需要特定版本，请获取 Elastic Beanstalk 支持的最新版本。有关受支持版本的列表，请参阅 AWS Elastic Beanstalk 平台文档中的 Go。

请从 https://golang.org/doc/install 下载 Go。

安装 AWS SDK for Go

如果您需要在应用程序中管理 AWS 资源，请使用以下命令安装 AWS SDK for Go。

```
$ go get github.com/aws/aws-sdk-go
```

有关更多信息，请参阅 AWS SDK for Go。

使用 Elastic Beanstalk Go 平台

您可以使用 AWS Elastic Beanstalk 来运行、生成和配置基于 Go 的应用程序。对于简单的 Go 应用程序，有两种部署方式：

- 通过根目录中名为 `application.go` 的源文件提供源包，该文件包含应用程序的主程序包。Elastic Beanstalk 使用以下命令生成二进制文件：

  ```
  go build -o bin/application application.go
  ```

应用程序生成后，Elastic Beanstalk 在端口 5000 上启动它。

- 通过名为 `application` 的二进制文件提供源包。此二进制文件可位于源包的根目录或源包的 `bin/` 目录中。如果您将 `application` 二进制文件同时放在这两个位置，Elastic Beanstalk 将使用 `bin/` 目录中的文件。

  Elastic Beanstalk 在端口 5000 上启动此应用程序。

在这两种情况下，利用 Go 1.11 或更高版本，您还可以在名为 `go.mod` 的文件中提供模块要求。有关更多信息，请参阅 Go 博客中的迁移到 Go 模块。

对于更复杂的 Go 应用程序，有两种部署方式：

- 提供包含应用程序源文件的源包，同时提供 Buildfile 和 Procfile。Buildfile 包含用于生成应用程序的命令，Procfile 包含用于运行应用程序的指令。
- 提供包含应用程序二进制文件的源包，同时提供 Procfile。Procfile 包含用于运行应用程序的指令。

Go 平台包括一个代理服务器，用于提供静态资产并将流量转发到您的应用程序。您可以扩展或覆盖默认代理配置，以适应高级方案。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅 the section called “扩展 Linux 平台”。

配置 Go 环境

通过 Go 平台设置，您可以微调 Amazon EC2 实例的行为。您可以使用 Elastic Beanstalk 控制台编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，并配置应用程序可从环境中读取的变量。

在 Elastic Beanstalk 控制台中配置 Go 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

日志选项

“日志选项”部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

环境属性

在环境属性部分，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。环境属性会以密钥值对的形式传递到应用程序。

在运行于 Elastic Beanstalk 中的 Go 环境内，可通过使用 `os.Getenv` 函数访问环境变量。例如，您可以使用以下代码将名为 `API_ENDPOINT` 的属性读取到某个变量：

```
endpoint := os.Getenv("API_ENDPOINT")
```

参阅 环境属性和其他软件设置 了解更多信息。

Go 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间中。

Go 平台不定义任何特定于平台的命名空间。您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间将代理配置为提供静态文件。有关详细信息和示例，请参阅the section called “静态文件”。

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

Amazon Linux AMI（在 Amazon Linux 2 之前）的 Go 平台

如果您的 Elastic Beanstalk Go 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

Go 配置命名空间 — Amazon Linux AMI (AL1)

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间中。

> **Note**
>
> 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。

除了所有平台支持的命名空间外，Amazon Linux AMI Go 平台还支持一个特定于平台的配置命名空间。使用 `aws:elasticbeanstalk:container:golang:staticfiles` 命名空间可以定义一些选项，通过它们将您的 Web 应用程序路径映射到应用程序源包中包含静态内容的文件夹。

例如，该配置文件告诉代理服务器在路径 `/images` 上的 `staticimages` 文件夹中提供文件：

Example .ebextensions/go-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:golang:staticfiles:
    /html: statichtml
    /images: staticimages
```

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

使用 Procfile 配置应用程序进程

要指定用于启动 Go 应用程序的自定义命令，请在源包根目录中包含一个名为 `Procfile` 的文件。

有关编写和使用 `Procfile` 的详细信息，请展开the section called “扩展 Linux 平台”中的 Buildfile 和 Procfile 部分 。

Example Procfile

```
web: bin/server
queue_process: bin/queue_processor
foo: bin/fooapp
```

您必须调用主应用程序 `web`，并在 `Procfile` 中将其列为第一个命令。Elastic Beanstalk 在环境的根 URL 上公开主 `web` 应用程序；例如，`http://my-go-env.elasticbeanstalk.com`。

Elastic Beanstalk 还会运行名称没有 `web_` 前缀的任何应用程序，但这些应用程序在您的实例外部不可用。

Elastic Beanstalk 希望从 `Procfile` 运行的进程一直运行。Elastic Beanstalk 会监控这些应用程序并重启所有终止的进程。对于短期运行的进程，请使用 Buildfile 命令。

在 Amazon Linux AMI（在 Amazon Linux 2 之前）上使用 Procfile

如果您的 Elastic Beanstalk Go 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。

- 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

端口传递 — Amazon Linux AMI (AL1)

> **Note**
>
> 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。

Elastic Beanstalk 配置 nginx 代理，使其在应用程序的 PORT 环境属性中指定的端口号上转发对应用程序的请求。您的应用程序应始终侦听该端口。您可以通过在应用程序中调用 `os.Getenv("PORT")` 方法访问此变量。

Elastic Beanstalk 使用 PORT 环境属性中指定的端口号作为 `Procfile` 中第一个应用程序的端口，然后对 `Procfile` 中的每个后续应用程序的端口号递增 100。如果未设置 PORT 环境属性，则 Elastic Beanstalk 使用 5000 作为初始端口。

在前面的示例中，`web` 应用程序的 PORT 环境属性为 5000，`queue_process` 应用程序使用 5100，`foo` 应用程序使用 5200。

您可以通过 aws:elasticbeanstalk:application:environment 命名空间设置 PORT 选项从而指定初始端口，如下例所示。

```
option_settings:
  - namespace:  aws:elasticbeanstalk:application:environment
    option_name:  PORT
    value:  <first_port_number>
```

有关为应用程序设置环境属性的更多信息，请参阅 选项设置。

使用 Buildfile 在服务器上生成可执行文件

要为您的 Go 应用程序指定自定义的生成和配置命令，请在源包根目录中包含一个名为 `Buildfile` 的文件。文件名区分大小写。`Buildfile` 应采用以下格式：

```
<process_name>: <command>
```

Buildfile 中的命令必须符合以下正则表达式：^[A-Za-z0-9_]+:\s*.+$。

Elastic Beanstalk 不会监控通过 Buildfile 运行的应用程序。对于短期运行并在完成任务后终止的命令，请使用 Buildfile。对于长期运行、不应退出的应用程序进程，请使用 Procfile。

在下面的 Buildfile 示例中，build.sh 是位于源包根目录的 Shell 脚本：

```
make: ./build.sh
```

Buildfile 中的所有路径都是源包根目录的相对路径。如果您事先知道这些文件在实例上的位置，则可以在 Buildfile 中使用绝对路径。

配置反向代理

Elastic Beanstalk 使用 nginx 作为反向代理，将应用程序映射到端口 80 上的 Elastic Load Balancing 负载均衡器。Elastic Beanstalk 提供一个默认 nginx 配置，您可以扩展该配置，也可以使用您自己的配置完全覆盖该配置。

默认情况下，Elastic Beanstalk 将 nginx 代理配置为通过端口 5000 向您的应用程序转发请求。您可以覆盖默认端口，方法是将 PORT 环境属性设置为主应用程序侦听的端口。

> **Note**
>
> 应用程序侦听的端口不会影响 nginx 服务器为了从负载均衡器接收请求而侦听的端口。

在平台版本上配置代理服务器

所有 AL2023/AL2 平台都支持统一的代理配置功能。有关在运行 AL2023/AL2 的平台版本上配置代理服务器的更多信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

在 Amazon Linux AMI（在 Amazon Linux 2 之前）上配置代理

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。

- 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

如果您的 Elastic Beanstalk Go 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的信息。

扩展并覆盖默认代理配置 — Amazon Linux AMI (AL1)

Elastic Beanstalk 使用 nginx 作为反向代理，将应用程序映射到端口 80 上的负载均衡器。如果您想提供自己的 nginx 配置，可以覆盖 Elastic Beanstalk 提供的默认配置，方法是在源包中包含 `.ebextensions/nginx/nginx.conf` 文件。如果此文件存在，Elastic Beanstalk 会使用它来替代默认的 nginx 配置文件。

如果除 `nginx.conf` `http` 块中的指令外，您还想包含其他指令，可以在源包的 `.ebextensions/nginx/conf.d/` 目录中提供其他配置文件。此目录中的所有文件都必须采用 `.conf` 扩展名。

要利用 Elastic Beanstalk 提供的功能（例如增强型运行状况报告和监控、自动应用程序映射和静态文件），您必须在 nginx 配置文件的 `server` 块中包含下面一行：

```
include conf.d/elasticbeanstalk/*.conf;
```

在 Elastic Beanstalk 上创建和部署 Java 应用程序

AWS Elastic Beanstalk 支持适用于 Java 应用程序的两个平台。

- Tomcat – 基于 Apache Tomcat 的平台，是一种开源 Web 容器，适用于使用 Java servlet 和 JavaServer 页面 (JSP) 处理 HTTP 请求的应用程序。Tomcat 通过提供多线程、声明性安全配置和丰富的自定义功能来帮助开发 Web 应用程序。Elastic Beanstalk 为 Tomcat 当前的每个主要版本提供平台分支。有关更多信息，请参阅Tomcat 平台。
- Java SE – 适用于不使用 Web 容器或使用 Tomcat 以外的其他容器（如 Jetty 或 GlassFish）的应用程序的平台。您可以在部署到 Elastic Beanstalk 的源包中包含应用程序所使用的任何库 Java 存档 (JAR)。有关更多信息，请参阅Java SE 平台。

Tomcat 和 Java SE 平台的最新分支都基于 Amazon Linux 2 及更高版本，并使用 Corretto（AWS Java SE 发行版）。平台列表中这些分支的名称包括 Corretto 一词而不是 Java，例如 `Corretto 11 with Tomcat 8.5`。

有关当前平台版本的列表，请参阅 AWS Elastic Beanstalk 平台指南中的 Tomcat 和 Java SE。

AWS提供了多种用于处理 Java 和 Elastic Beanstalk 的工具。无论您选择何种平台分支，都可以使用 AWS SDK for Java 从 Java 应用程序内使用其他 AWS 服务。AWS SDK for Java 是一组库，使您可以从应用程序代码中使用 AWS API，而无需从头开始编写原始 HTTP 调用。

如果您使用 Eclipse 集成开发环境 (IDE) 开发 Java 应用程序，还可以得到 AWS Toolkit for Eclipse。AWS Toolkit for Eclipse 是一种开源插件，您可用来在 Eclipse IDE 中管理 AWS 资源，包括 Elastic Beanstalk 应用程序和环境。

如果您更习惯使用命令行，请安装 Elastic Beanstalk 命令行界面 (EB CLI)，并使用它从命令行创建、监控和管理您的 Elastic Beanstalk 环境。如果您为应用程序运行多个环境，则 EB CLI 可与 Git 集成，以使您可以将每个环境与不同 Git 分支关联。

本章中的主题假设您对 Elastic Beanstalk 环境有所了解。如果您以前未使用过 Elastic Beanstalk，请尝试使用入门教程以了解基本知识。

主题

开始在 Elastic Beanstalk 上使用 Java

要开始在 AWS Elastic Beanstalk 上使用 Java 应用程序，您只需一个应用程序源包，将其作为第一个应用程序版本上传并部署到环境中。创建环境时，Elastic Beanstalk 会分配运行可扩展 Web 应用程序所需的所有 AWS 资源。

使用示例 Java 应用程序启动环境

Elastic Beanstalk 为每个平台提供单页示例应用程序以及更复杂的示例，这些示例展示了 Amazon RDS 等其他 AWS 资源的使用情况以及特定于语言或平台的功能和 API。

单一页面示例是您在创建环境时获得的相同代码，无需提供您自身的源代码。更复杂的示例托管在 GitHub 上，可能需要在部署到 Elastic Beanstalk 环境之前进行编译或构建。

示例

名称	支持的版本	环境类型	源	描述
Tomc 页 ）	所有 Tomcat（带 Corretto）平台分支	Web 服务器 工作线程	tomcat.zip	配置为仅在网站根目录中显示一个页面 (`index.jsp`) 的 Tomcat Web 应用程序。 对于工作线程环境，此示例包含一个 `cron.yaml` 文件，该文件配置一个每分钟调用一次 `scheduled.jsp` 的计划任务。调用 `scheduled.jsp` 时，它会写入 `/tmp/sample-app.log` 处的日志文件。最后，在 `.ebextensions` 中包含一个配置文件，该文件在您请求环境日志时将日志从 `/tmp/` 复制到由 Elastic Beanstalk 读取的位置。 如果您在运行此示例的环境中启用 X-Ray 集成，则应用程序会显示有关 X-Ray 的更多内容，并提供用于生成调试信息（可在 X-Ray 控制台中查看）的选项。
Corre 单 页 ）	Corretto 11 Corretto 8	Web 服务器	corretto.zip	使用 `Buildfile` 和 `Procfile` 配置文件的 Corretto 应用程序。 如果您在运行此示例的环境中启用 X-Ray 集成，则应用程序会显示有关 X-Ray 的更多内

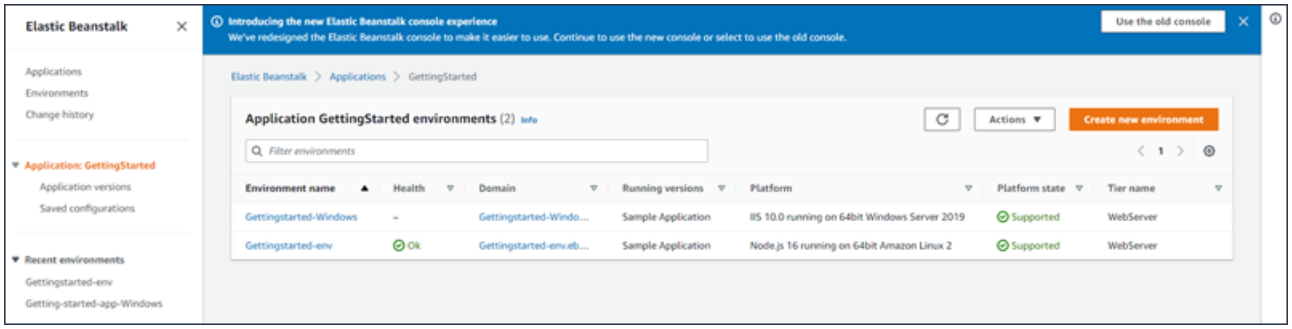
名称	支持的版本	环境类型	源	描述
				容，并提供用于生成调试信息（可在 X-Ray 控制台中查看）的选项。
Score	Java 8	Web 服务器	克隆 GitHub.com 的报告	Scorekeep 是一种 RESTful Web API，它借助 Spring 框架提供用于创建和管理用户、会话及游戏的接口。此 API 与通过 HTTP 使用它的 Angular 1.5 Web 应用程序捆绑在一起。 应用程序使用 Java SE 平台的功能下载依赖项和用作构建基础的实例，从而最小化源包的大小。此外，应用程序还包含覆盖默认配置的 nginx 配置文件，以通过代理在端口 80 上静态地为前端 Web 应用程序提供服务，并将针对 `/api` 下路径的请求路由到在 `localhost:5000` 上运行的 API。 Scorekeep 还包含一个 `xray` 分支，它演示了使用 AWS X-Ray 检测 Java 应用程序的方法。它演示了使用 servlet 筛选条件进行传入 HTTP 请求检测、自动和手动 AWS SDK 客户端检测、记录器配置以及传出 HTTP 请求和 SQL 客户端的检测方法。 请参阅自述文件获取说明，或跟随 AWS X-Ray 入门教程演练用 X-Ray 检测应用程序的过程。

名称	支持的版本	环境类型	源	描述
是否有 Snak	使用 Java 8 的 Tomcat 8	Web 服务器	克隆 GitHub.com 的报告	是否有 Snake？是 Tomcat Web 应用程序，显示 Elastic Beanstalk 配置文件、Amazon RDS、JDBC、PostgreSQL、Servlet、JSP、Simple Tag Support、Tag Files、Log4J、Bootstrap 和 Jackson 的使用。 此项目的源代码包括最低要求构建脚本，后者将小服务程序和模型汇编入类文件，并将所需文件封装入您可部署至 Elastic Beanstalk 环境的 Web 存档。请参阅项目存储库中的自述文件，浏览完整说明。
Locus Load Gene	Java 8	Web 服务器	克隆 GitHub.com 的报告	您可用于对不同 Elastic Beanstalk 环境中运行的另一个 Web 应用程序进行负载测试的 Web 应用程序。显示 `Buildfile` 和 `Procfile` 文件、DynamoDB 以及 Locust（一种开放源负载测试工具）的使用。

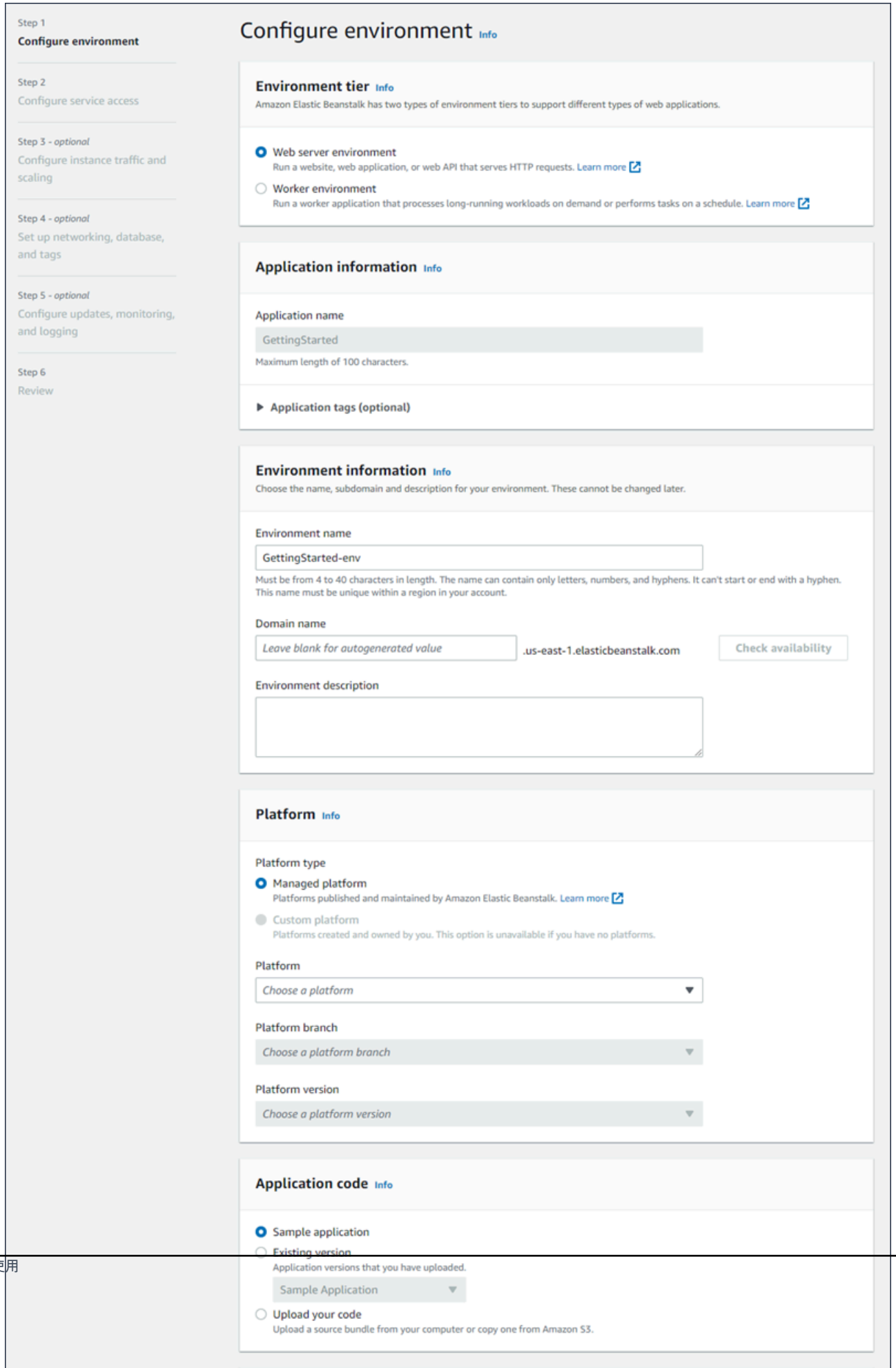
按照以下步骤下载任何示例应用程序并将其部署到 Elastic Beanstalk：

使用示例应用程序启动环境 (控制台)

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择应用程序，然后在列表中选择现有应用程序的名称或创建一个。
3. 在应用程序概述页面上，选择 Create new environment（创建新环境）。



这将启动 Create environment（创建环境）向导。该向导提供了一组创建新环境的步骤。


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Configure instance traffic and scaling
Step 4 - optional
Set up networking, database, and tags
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure environment Info
Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more
Application information Info
Application name
GettingStarted
Maximum length of 100 characters.
Application tags (optional)
Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.
Environment name
GettingStarted-env
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.
Domain name
Leave blank for autogenerated value
.us-east-1.elasticbeanstalk.com
Check availability
Environment description
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Choose a platform
Platform branch
Choose a platform branch
Platform version
Choose a platform version
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Sample Application
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

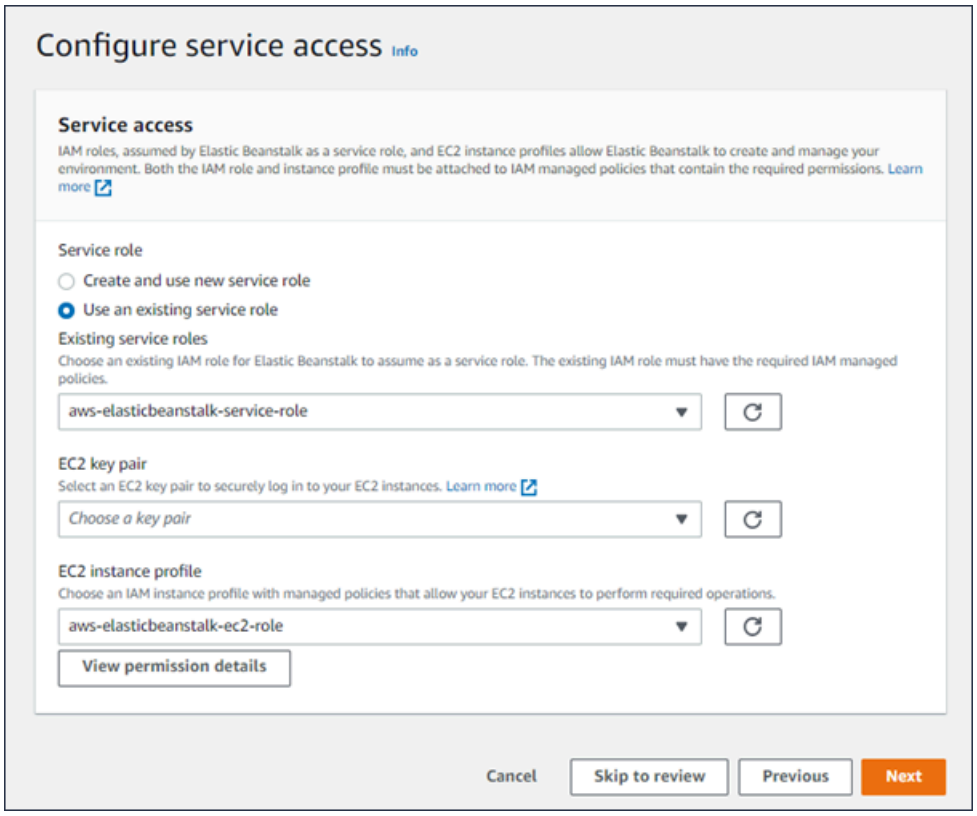
4. 对于环境层，选择 Web server environment（Web 服务器环境）或 Worker environment（工作线程环境）环境层。环境的层创建后无法更改。

> **Note**
>
> .NET on Windows Server 平台不支持工作线程环境层。

5. 对于平台，选择与应用程序使用的语言匹配的平台和平台分支。

> **Note**
>
> Elastic Beanstalk 支持列出的大多数平台的多个版本。默认情况下，此控制台将为您选择的平台和平台分支选择推荐版本。如果您的应用程序需要其他版本，您可以在此处选择该版本。有关支持的平台版本的信息，请参阅the section called “支持的平台”。

6. 对于应用程序代码，选择示例应用程序。
7. 对于 Configuration presets（配置预设），选择 Single instance（单一实例）。
8. 选择 Next（下一步）。
9. 这时将显示配置服务访问权限页面。



10. 对于服务角色，请选择使用现有的服务角色。
11. 下面我们将重点介绍 EC2 实例配置文件下拉列表。此下拉列表中显示的值可能因您的账户之前是否创建过新环境而异。

    根据列表中显示的值，选择以下选项中的一个。

    - 如果在下拉列表中显示有 `aws-elasticbeanstalk-ec2-role`，请从 EC2 实例配置文件下拉列表中将其选中。
    - 如果列表中显示的是其他值，并且这是您环境的默认 EC2 实例配置文件，请从 EC2 实例配置文件下拉列表中选中该值。
    - 如果 EC2 实例配置文件下拉列表未显示任何可供选择的值，请按下面为 EC2 实例配置文件创建 IAM 角色的过程操作。

      完成为 EC2 实例配置文件创建 IAM 角色中的步骤，以创建一个之后为 EC2 实例配置文件选择的 IAM 角色。然后返回此步骤。

      现在您已创建了一个 IAM 角色并刷新了列表，该角色将在下拉列表中显示为一个选项。从 EC2 实例配置文件下拉列表中选中您刚刚创建的 IAM 角色。
12. 在 Configure service access（配置服务访问）页面上选择 Skip to Review（跳至审核）。

    这样做将选择此步骤的默认值，并跳过可选步骤。
13. Review（审核）页面将显示所有选择的摘要。

    要进一步自定义您的环境，请在包含要配置的任何项目的步骤旁边选择 Edit（编辑）。只能在创建环境期间设置下列选项：

    - 环境名称
    - 域名
    - 平台版本
    - 处理器
    - VPC
    - 套餐

    可在环境创建后更改下列设置，但它们需要配置新实例或其他资源并且应用更改可能需要很长的时间：

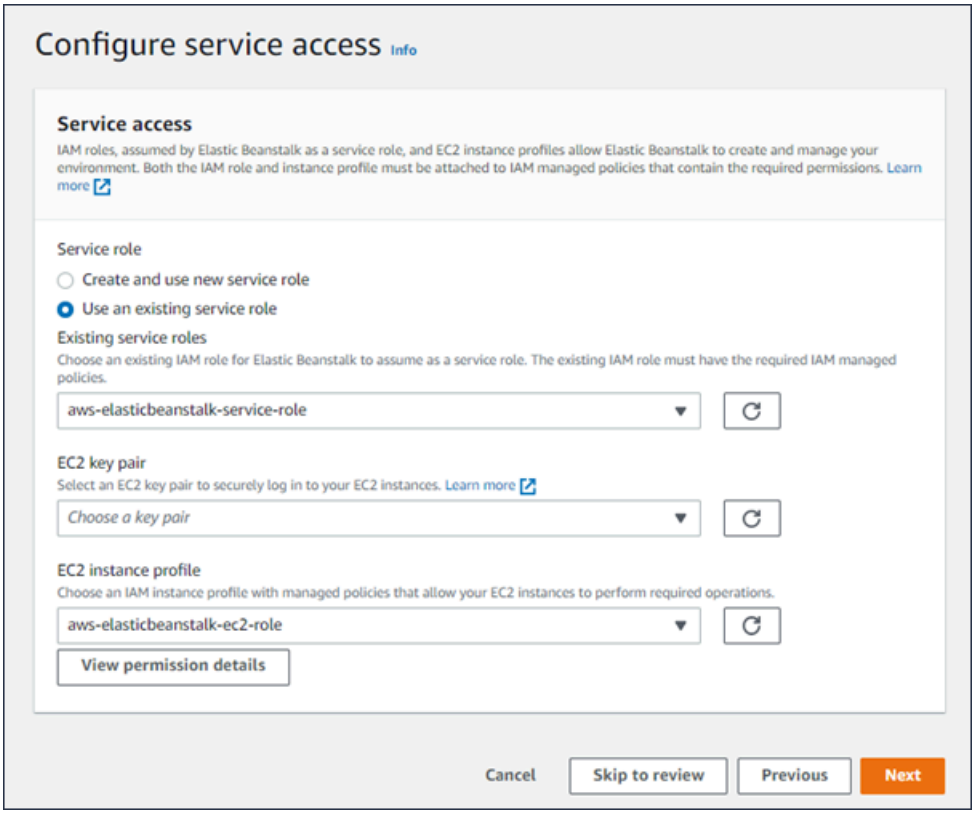
    - 实例类型、根卷、密钥对和 AWS Identity and Access Management (IAM) 角色

- 内部 Amazon RDS 数据库
- 负载均衡器

有关所有可用设置的详细信息，请参阅创建新环境向导。

14. 选择页面底部的 Submit（提交）以初始化新环境的创建。

为 EC2 实例配置文件创建 IAM 角色



创建用于 EC2 实例配置文件选择的 IAM 角色

1. 选择查看权限详细信息。这将在 EC2 实例配置文件下拉列表下显示。

   这时会显示一个名为查看实例配置文件权限的模态窗口。此窗口将列出您需要附加到所创建的新 EC2 实例配置文件的托管式配置文件。此外还提供了一个用于启动 IAM 控制台的链接。

2. 选择窗口顶部显示的 IAM 控制台链接。
3. 请在 IAM 控制台的导航窗格中，选择 Roles（角色）。
4. 选择 Create role（创建角色）。
5. 在可信实体类型下，选择 AWS 服务。

6. 在 Use case（使用案例）下，选择 EC2。
7. 选择 Next（下一步）。
8. 附加适当的托管式策略。滚动查看实例配置文件权限模式窗口，以查看托管式策略。这些策略还将在此处列出：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 选择 Next（下一步）。
10. 输入角色的名称。
11. （可选）将标签添加到角色。
12. 选择 Create role（创建角色）。
13. 返回已打开的 Elastic Beanstalk 控制台窗口。
14. 关闭查看实例配置文件权限模态窗口。

> **Important**
>
> 不要关闭显示 Elastic Beanstalk 控制台的浏览器页面。

15. 选择 EC2 实例配置文件下拉列表旁边的  （刷新）。

    这将刷新下拉列表，以确保您刚刚创建的角色会在下拉列表中显示。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或完全不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。

在您部署了一两个示例应用程序并准备好开始在本地开发和运行 Java 应用程序以后，请参阅下一节以使用所有需要的工具和库设置 Java 开发环境。

设置 Java 开发环境

设置 Java 开发环境以在本地测试应用程序，然后再将之部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

有关适用于所有语言的常见设置步骤和工具，请参阅配置您的开发计算机。

小节目录

安装 Java 开发工具包

安装 Java 开发工具包 (JDK)。如果您没有特别的要求，请获取最新版本。从 oracle.com 下载 JDK

JDK 包括 Java 编译器，您可以使用它将源文件构建为可在 Elastic Beanstalk Web 服务器上执行的类文件。

安装 Web 容器

如果您还没有其他 Web 容器或框架，请安装合适的 Tomcat 版本：

- 下载 Tomcat 8（需要 Java 7 或更高版本）
- 下载 Tomcat 7（需要 Java 6 或更高版本）

下载库

Elastic Beanstalk 平台默认包含几个库。请下载您的应用程序要用到的库，并将它们保存到要在应用程序源包中部署的项目文件夹中。

如果您在本地安装了 Tomcat，可以从安装文件夹复制 servlet API 和 JavaServer 页面 (JSP) API 库。如果部署到 Tomcat 平台版本，则无需在源包中包含这些文件，但您的 `classpath` 中需要包含它们以便编译使用它们的任何类。

JUnit、Google Guava 和 Apache Commons 提供了几个很有用的库。请访问其主页了解更多信息：

- 下载 JUnit
- 下载 Google Guava
- 下载 Apache Commons

安装 AWS SDK for Java

如果您需要在应用程序中管理 AWS 资源，请安装 AWS SDK for Java。例如，借助 AWS SDK for Java，您可以使用 Amazon DynamoDB (DynamoDB) 跨多个 Web 服务器共享 Apache Tomcat 应用程序的会话状态。有关更多信息，请参阅 AWS SDK for Java 文档中的使用 Amazon DynamoDB 管理 Tomcat 会话状态。

有关更多信息和安装说明，请访问 AWS SDK for Java 主页。

安装 IDE 或文本编辑器

集成开发环境 (IDE) 提供了便于应用程序开发的大量功能。如果您还没使用 IDE 进行过 Java 开发，请尝试 Eclipse 和 IntelliJ，看哪个更适合您。

- 安装面向 Java EE 开发人员的 Eclipse IDE
- 安装 IntelliJ

> **Note**
>
> IDE 可以将您可能不希望提交到源代码控制的文件添加到项目文件夹中。要防止将这些文件提交到源代码控制，请使用 `.gitignore` 或您的源代码控制工具的同类功能。

如果您只是希望开始编码而不需要所有 IDE 功能，请考虑安装 Sublime Text。

安装 AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 是适用于 Eclipse Java IDE 的开源插件，能够方便开发人员使用 AWS 开发、调试和部署 Java 应用程序。有关安装说明，请访问 AWS Toolkit for Eclipse 主页。

使用 Elastic Beanstalk Tomcat 平台

> **Important**
>
> AWS Elastic Beanstalk 从适用于 Amazon Linux 1 和 Amazon Linux 2 的 Tomcat 平台中的 Amazon Linux 默认程序包存储库安装 Log4j。Amazon Linux 1 和 Amazon Linux 2 存储库中可用的 Log4j 版本在其原定设置配置中不受 CVE-2021-44228 或 CVE-2021-45046 的影响。
> 如果您对应用程序使用的 log4j 进行了配置更改，或者安装了 log4j 的更新版本，我们建议您采取措施更新应用程序的代码以缓解此问题。
> 出于谨慎考虑，Elastic Beanstalk 在 2021 年 12 月 21 日的 Amazon Linux 平台版本中发布了使用最新 Amazon Linux 默认程序包存储库的新平台版本，其中包括 Log4j 热补丁 JDK。如果您已将 log4j 安装自定义为应用程序依赖项，我们建议您升级到最新的 Elastic Beanstalk 平台版本，以缓解 CVE-2021-44228 或 CVE-2021-45046 问题。根据常规更新实践，您还可以启用自动化的托管式更新。
> 有关适用于 Amazon Linux 的安全相关软件更新的更多信息，请访问 Amazon Linux 安全中心。

AWS Elastic Beanstalk Tomcat 平台是一组平台版本，用于可以在 Tomcat Web 容器中运行的 Java Web 应用程序。Tomcat 在 nginx 代理服务器后面运行。每个平台分支都对应于 Tomcat 的一个主要版本（如 Java 8 with Tomcat 8）。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

Elastic Beanstalk Tomcat 平台包括一个可将请求转发到应用程序的反向代理。您可以使用配置选项将代理服务器配置为从您的源代码中的某个文件夹的静态资产提供服务，以减少应用程序的负载。有关高级方案，您可以在源包中包括您自己的 `.conf` 文件，以扩展 Elastic Beanstalk 代理配置或完全重写它。

> **Note**
>
> Elastic Beanstalk 支持将 nginx（默认）和 Apache HTTP Server 作为 Tomcat 平台上的代理服务器。如果您的 Elastic Beanstalk Tomcat 环境使用 Amazon Linux AMI 平台分支（在 Amazon Linux 2 之前），则您还可以选择使用 Apache HTTP Server 2.2 版。Apache（最新版本）是这些较旧平台分支上的默认版本。

2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

您必须在具有特定结构的 Web 应用程序存档 (WAR) 文件中打包 Java 应用程序。有关所需结构及其如何与项目目录结构相关的信息，请参阅项目文件夹结构设置。

要在同一 Web 服务器上运行多个应用程序，您可以将多个 WAR 文件捆绑到同一个源包中。多 WAR 源包中的每个应用程序在根路径运行 (`ROOT.war` 在 *`myapp`*`.elasticbeanstalk.com/` 运行)，或者在根路径的直接下级路径运行 (`app2.war` 在 *`myapp`*`.elasticbeanstalk.com/`*`app2`*`/` 运行)，具体通过 WAR 的名称确定。在单 WAR 源包中，应用程序始终在根路径运行。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅the section called “扩展 Linux 平台”。

主题

- 配置 Tomcat 环境
- Tomcat 配置命名空间
- 捆绑用于 Tomcat 环境的多个 WAR 文件
- 项目文件夹结构设置
- 配置 Tomcat 环境的代理服务器

配置 Tomcat 环境

Elastic Beanstalk Tomcat 平台除了提供所有平台都具有的标准选项之外，还提供了一些特定于平台的选项。这些选项可用于配置在环境的 Web 服务器上运行的 Java 虚拟机 (JVM)，以及定义向应用程序提供信息配置字符串的系统属性。

您可以使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，并配置应用程序可从环境中读取的变量。

在 Elastic Beanstalk 控制台中配置 Tomcat 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

容器选项

您可以指定以下特定于平台的选项：

- Proxy server（代理服务器）– 要在环境实例上使用的代理服务器。默认情况下，使用 nginx。

JVM 容器选项

Java 虚拟机 (JVM) 中的堆大小决定了在垃圾收集发生之前，您的应用程序中多少个对象可以在内存中创建。您可以修改初始 JVM 堆大小 (`-Xms option`) 和最大 JVM 堆大小 (`-Xmx`) 选项。初始堆大小越大，垃圾收集发生之前可以创建的对象就越多，但这也意味着垃圾收集器将花更长的时间去压缩堆。最大堆大小指定了在占用大量内存的活动期间扩展堆时，JVM 可分配的最大内存量。

> **Note**
>
> 可用内存取决于 Amazon EC2 实例类型。有关可用于您的 Elastic Beanstalk 环境的 EC2 实例类型的更多信息，请参阅适用于 Linux 实例的 Amazon Elastic Compute Cloud 用户指南 中的实例类型。

持久代 是 JVM 堆的一部分，可存储类定义和关联的元数据。要修改持久代的大小，请在最大 JVM PermGen 大小 (`-XX:MaxPermSize`) 选项中键入新的大小。此设置仅适用于 Java 7 和更低版本。该选项已在 JDK 8 中弃用，并由 MaxMetaspace 大小 (`-XX:MaxMetaspaceSize`) 选项所取代。

> **Important**
>
> JDK 17 不再支持 Java `-XX:MaxPermSize` 选项。在 Elastic Beanstalk 平台分支（带 Corretto 17）上运行的环境中使用该选项会导致错误。Elastic Beanstalk 于 2023 年 7 月 13 日发布了首个运行 Tomcat 的平台分支（带 Corretto 17）。
> 有关详细信息，请参阅以下资源：
>
> - Oracle Java 文档网站：移除的 Java 选项
> - Oracle Java 文档网站：其他注意事项中的类元数据部分

有关 Elastic Beanstalk 平台及其组件的更多信息，请参阅《AWS Elastic Beanstalk 平台指南》中的支持的平台。

日志选项

日志选项部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

环境属性

在 Environment Properties（环境属性）部分中，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。环境属性会以密钥值对的形式传递到应用程序。

Tomcat 平台为 Tomcat 环境定义了一个名为 `JDBC_CONNECTION_STRING` 的占位符属性，用于传递连接外部数据库的连接字符串。

> **Note**
>
> 如果您将 RDS 数据库实例附加到您的环境，请根据 Elastic Beanstalk 提供的 Amazon Relational Database Service (Amazon RDS) 环境属性动态构造 JDBC 连接字符串。JDBC_CONNECTION_STRING 只适用于不是使用 Elastic Beanstalk 预配置的数据库实例。
>
> 有关配合使用 Amazon RDS 和 Java 应用程序的更多信息，请参阅向 Java 应用程序环境中添加 Amazon RDS 数据库实例。

在运行于 Elastic Beanstalk 中的 Tomcat 环境内，可通过使用 `System.getProperty()` 访问环境变量。例如，可以使用以下代码将名为 `API_ENDPOINT` 的属性读取到某个变量。

```
String endpoint = System.getProperty("API_ENDPOINT");
```

参阅 环境属性和其他软件设置 了解更多信息。

Tomcat 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

除了所有 Elastic Beanstalk 环境支持的选项之外，Tomcat 平台还支持以下命名空间中的选项：

- `aws:elasticbeanstalk:container:tomcat:jvmoptions` – 修改 JVM 设置。此命名空间中的选项对应于管理控制台中的选项，如下所示：
  - `Xms` – JVM 命令行选项
  - `JVM Options` – JVM 命令行选项
- `aws:elasticbeanstalk:environment:proxy` – 选择环境的代理服务器。

以下示例配置文件演示了特定于 Tomcat 的配置选项的使用。

Example .ebextensions/tomcat-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:tomcat:jvmoptions:
    Xms: 512m
    JVM Options: '-Xmn128m'
```

```
  aws:elasticbeanstalk:application:environment:
    API_ENDPOINT: mywebapi.zkpexsjtmd.us-west-2.elasticbeanstalk.com
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
```

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

Amazon Linux AMI（在 Amazon Linux 2 之前）的 Tomcat 平台

如果您的 Elastic Beanstalk Tomcat 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

Tomcat 配置命名空间 — Amazon Linux AMI (AL1)

Tomcat Amazon Linux AMI 平台支持以下命名空间中的附加选项：

- `aws:elasticbeanstalk:container:tomcat:jvmoptions` – 除了本页前面提到的这个命名空间的选项之外，较旧的 Amazon Linux AMI 平台版本还支持：
  - `XX:MaxPermSize` – 最大 JVM 持久生成大小
- `aws:elasticbeanstalk:environment:proxy` – 除了选择代理服务器之外，还配置响应压缩。

以下示例配置文件展示了代理命名空间配置选项的使用。

Example .ebextensions/tomcat-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    GzipCompression: 'true'
```

```
    ProxyServer: nginx
```

包含 Elastic Beanstalk 配置文件 — Amazon Linux AMI (AL1)

要部署 `.ebextensions` 配置文件，请将其包含在您的应用程序源中。对于单个应用程序，将通过运行以下命令将 `.ebextensions` 添加到压缩的 WAR 文件：

Example

```
zip -ur your_application.war .ebextensions
```

对于需要多个 WAR 文件的应用程序，如需更多说明，请参阅捆绑用于 Tomcat 环境的多个 WAR 文件。

捆绑用于 Tomcat 环境的多个 WAR 文件

如果您的 Web 应用程序包含多个 Web 应用程序组件，您可以在一个环境中运行多个组件，而不必在单独的环境中运行每个组件，从而简化部署和降低运营成本。此策略对无需大量资源的轻型应用程序以及开发和测试环境很有效。

要将多个 Web 应用程序部署到您的环境，请将每个组件的 Web 应用程序存档 (WAR) 文件合并到单个源包中。

要创建包含多个 WAR 文件的应用程序源包，请使用以下结构来组织 WAR 文件。

```
MyApplication.zip
### .ebextensions
### .platform
### foo.war
### bar.war
### ROOT.war
```

当您将包含多个 WAR 文件的源包部署到 AWS Elastic Beanstalk 环境时，每个应用程序均可从根域名的不同路径进行访问。前面的示例包含三个应用程序：`foo`、`bar` 和 `ROOT`。`ROOT.war` 是一个特殊文件名，它告知 Elastic Beanstalk 在根域中运行该应用程序，以便三个应用程序可在 `http://MyApplication.elasticbeanstalk.com/foo`、`http://MyApplication.elasticbeanstalk.com/bar` 和 `http://MyApplication.elasticbeanstalk.com` 中使用。

源包可以包括 WAR 文件、可选的 `.ebextensions` 文件夹和可选的 `.platform` 文件夹。有关这些可选配置文件夹的详细信息，请参阅the section called “扩展 Linux 平台”。

启动环境 (控制台)

1. 使用下面的预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced
2. 对于平台，请选择与应用程序使用的语言匹配的平台和平台分支，或者为基于容器的应用程序选择 Docker 平台。
3. 对于 Application code（应用程序代码），选择 Upload your code（上传代码）。
4. 选择 Local file (本地文件)，再选择 Choose file (选择文件)，然后打开源包。
5. 选择复查并启动。
6. 查看可用设置并选择 Create app (创建应用程序)。

有关创建源包的信息，请参阅创建应用程序源包。。

项目文件夹结构设置

要在部署到 Tomcat 服务器时正常工作，必须根据某些准则为编译的 Java Platform Enterprise Edition (Java EE) Web 应用程序存档（WAR 文件）设置正确的结构。项目目录不一定要遵循同样的标准，但为了简化编译和打包操作，最好采用同样的方式设置结构。如同 WAR 文件内容一样设置项目文件夹的结构，还有助您了解文件的关联方式以及它们在 Web 服务器上的行为方式。

在以下推荐使用的层次结构中，Web 应用程序的源代码置于 `src` 目录中，与生成脚本和它生成的 WAR 文件隔离。

```
~/workspace/my-app/
|-- build.sh            - Build script that compiles classes and creates a WAR
|-- README.MD           - Readme file with information about your project, notes
|-- ROOT.war            - Source bundle artifact created by build.sh
`-- src                 - Source code folder
    |-- WEB-INF         - Folder for private supporting files
    |   |-- classes     - Compiled classes
    |   |-- lib         - JAR libraries
    |   |-- tags        - Tag files
    |   |-- tlds        - Tag Library Descriptor files
    |   `-- web.xml     - Deployment Descriptor
    |-- com             - Uncompiled classes
    |-- css             - Style sheets
    |-- images          - Image files
    |-- js              - JavaScript files
```

```
    `-- default.jsp     - JSP (JavaServer Pages) webpage
```

src 文件夹内容与将进行打包并部署到服务器的内容匹配 (com 文件夹除外)。com 文件夹包含未编译的类 (.java 文件)。这些类需要编译并置于 WEB-INF/classes 目录中，以便可通过应用程序代码进行访问。

WEB-INF 目录包含未在 Web 服务器上公开提供的代码和配置。源目录根位置处的其他文件夹 (css、images 和 js) 在 Web 服务器上的对应路径中公开提供。

以下列示例与上面的项目目录相同，只是包含的文件和子目录更多。此示例项目包含简单的标签、模型和支持类以及一个用于 record 资源的 Java Server Page (JSP) 文件。它还包含一个用于引导的样式表和 JavaScript、一个默认 JSP 文件以及 404 错误对应的一个错误页面。

WEB-INF/lib 包含一个 Java 存档 (JAR) 文件，其中含有用于 PostgreSQL 的 Java 数据库连接 (JDBC) 驱动程序。WEB-INF/classes 为空，因为尚未编译类文件。

```
~/workspace/my-app/
|-- build.sh
|-- README.MD
|-- ROOT.war
`-- src
    |-- WEB-INF
    |   |-- classes
    |   |-- lib
    |   |   `-- postgresql-9.4-1201.jdbc4.jar
    |   |-- tags
    |   |   `-- header.tag
    |   |-- tlds
    |   |   `-- records.tld
    |   `-- web.xml
    |-- com
    |   `-- myapp
    |       |-- model
    |       |   `-- Record.java
    |       `-- web
    |           `-- ListRecords.java
    |-- css
    |   |-- bootstrap.min.css
    |   `-- myapp.css
    |-- images
    |   `-- myapp.png
    |-- js
```

```
    |   `-- bootstrap.min.js
    |-- 404.jsp
    |-- default.jsp
    `-- records.jsp
```

使用 Shell 脚本生成 WAR 文件

build.sh 是非常简单的 shell 脚本，它用于编译 Java 类、构造 WAR 文件并将该文件复制到 Tomcat webapps 目录以进行本地测试。

```
cd src
javac -d WEB-INF/classes com/myapp/model/Record.java
javac -classpath WEB-INF/lib/*:WEB-INF/classes -d WEB-INF/classes com/myapp/model/
Record.java
javac -classpath WEB-INF/lib/*:WEB-INF/classes -d WEB-INF/classes com/myapp/web/
ListRecords.java

jar -cvf ROOT.war *.jsp images css js WEB-INF
cp ROOT.war /Library/Tomcat/webapps
mv ROOT.war ../
```

在 WAR 文件中，将发现与上面示例中的 src 目录相同的结构 (不包括 src/com 文件夹)。jar 命令自动创建 META-INF/MANIFEST.MF 文件。

```
~/workspace/my-app/ROOT.war
|-- META-INF
|   `-- MANIFEST.MF
|-- WEB-INF
|   |-- classes
|   |   `-- com
|   |       `-- myapp
|   |           |-- model
|   |           |   `-- Records.class
|   |           `-- web
|   |               `-- ListRecords.class
|   |-- lib
|   |   `-- postgresql-9.4-1201.jdbc4.jar
|   |-- tags
|   |   `-- header.tag
|   |-- tlds
|   |   `-- records.tld
|   `-- web.xml
```

```
|-- css
|   |-- bootstrap.min.css
|   `-- myapp.css
|-- images
|   `-- myapp.png
|-- js
|   `-- bootstrap.min.js
|-- 404.jsp
|-- default.jsp
`-- records.jsp
```

使用 `.gitignore`

要避免将已编译的类文件和 WAR 文件提交到 Git 存储库，或避免在运行 Git 命令时显示有关这些文件的消息，请将相关文件类型添加到项目文件夹内一个名为 `.gitignore` 的文件中。

~/workspace/myapp/.gitignore

```
*.zip
*.class
```

配置 Tomcat 环境的代理服务器

Tomcat 平台使用 nginx（默认）或 Apache HTTP Server 作为反向代理，将实例上来自端口 80 的请求中继到在端口 8080 上侦听的 Tomcat Web 容器。Elastic Beanstalk 提供一个默认代理配置，您可以扩展该配置，也可以使用您自己的配置完全覆盖该配置。

在平台版本上配置代理服务器

所有 AL2023/AL2 平台都支持统一的代理配置功能。有关在运行 AL2023/AL2 的平台版本上配置代理服务器的更多信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

在 Amazon Linux AMI（在 Amazon Linux 2 之前）的 Tomcat 平台上配置代理

如果您的 Elastic Beanstalk Tomcat 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。

- 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

为您的 Tomcat 环境选择代理服务器 — Amazon Linux AMI (AL1)

默认情况下，基于 Amazon Linux AMI（在 Amazon Linux 2 之前）的 Tomcat 平台版本为代理使用 Apache 2.4。可以通过将配置文件包括在源代码中来选择使用 Apache 2.2 或 nginx。以下示例将 Elastic Beanstalk 配置为使用 nginx。

Example .ebextensions/nginx-proxy.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: nginx
```

从 Apache 2.2 迁移到 Apache 2.4 — Amazon Linux AMI (AL1)

如果应用程序是针对 Apache 2.2 开发的，请阅读此部分以了解有关迁移到 Apache 2.4 的信息。

从 Tomcat 平台版本 3.0.0 配置（随 2018 年 5 月 24 日的 Java with Tomcat 平台更新一起发布）开始，Apache 2.4 是 Tomcat 平台的默认代理。Apache 2.4 `.conf` 文件大多 (但不是全部) 向后兼容 Apache 2.2 的此类文件。Elastic Beanstalk 包含可在每个 Apache 版本中正常工作的默认 `.conf` 文件。如果应用程序未自定义 Apache 的配置 (如扩展并覆盖默认 Apache 配置 — Amazon Linux AMI (AL1)中所述)，它应准确无误地迁移到 Apache 2.4。

如果应用程序扩展或覆盖 Apache 的配置，可能必须进行一些更改才能迁移到 Apache 2.4。有关更多信息，请参阅 Apache 软件基金会 站点上的从 2.2 升级到 2.4。在成功迁移到 Apache 2.4 之前，可以通过在源代码中包括以下配置文件作为临时措施，选择将 Apache 2.2 与应用程序一起使用。

Example .ebextensions/apache-legacy-proxy.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache/2.2
```

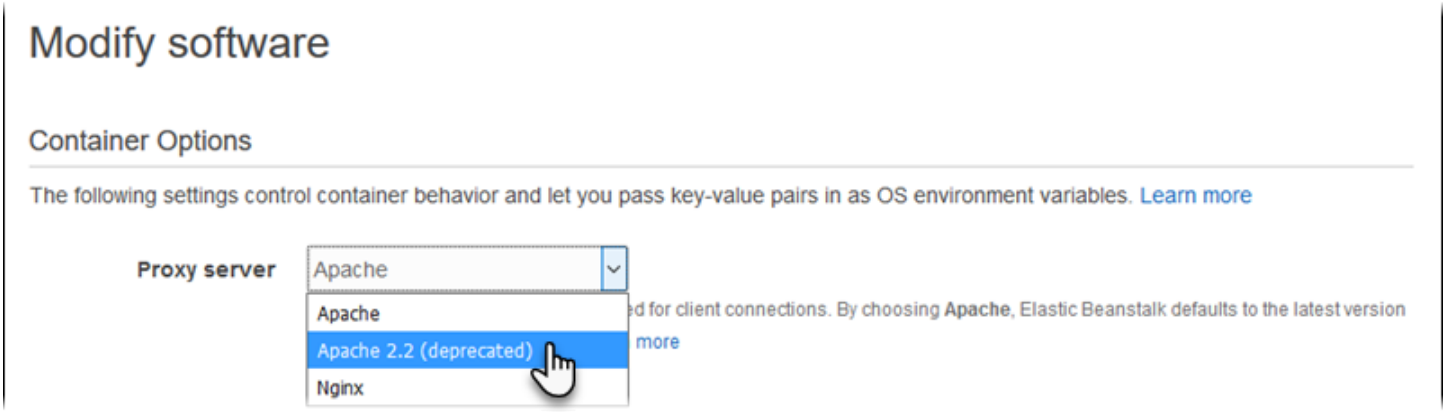
为了快速修复，您还可以在 Elastic Beanstalk 控制台中选择代理服务器。

在 Elastic Beanstalk 控制台中选择 Tomcat 环境中的代理

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 对于 Proxy server（代理服务器），请选择 `Apache 2.2 (deprecated)`。
6. 要保存更改，请选择页面底部的 Apply（应用）。



扩展并覆盖默认 Apache 配置 — Amazon Linux AMI (AL1)

您可以使用其他配置文件扩展 Elastic Beanstalk 默认 Apache 配置。也可以完全覆盖 Elastic Beanstalk 默认 Apache 配置。

> **Note**
>
> - 所有 Amazon Linux 2 平台都支持统一的代理配置功能。有关在运行 Amazon Linux 2 的 Tomcat 平台版本上配置代理服务器的详细信息，请展开the section called “扩展 Linux 平台”中的反向代理配置部分。
> - 如果您要将 Elastic Beanstalk 应用程序迁移到 Amazon Linux 2 平台，请务必阅读the section called “迁移到 AL2023/AL2” 中的信息。

要扩展 Elastic Beanstalk 默认 Apache 配置，请将 `.conf` 配置文件添加到应用程序源包中名为 `.ebextensions/httpd/conf.d` 的文件夹中。Elastic Beanstalk Apache 配置自动在此文件夹中包括 `.conf` 文件。

```
~/workspace/my-app/
|-- .ebextensions
|   -- httpd
|      -- conf.d
|         -- myconf.conf
|         -- ssl.conf
-- index.jsp
```

例如，以下 Apache 2.4 配置将在端口 5000 上添加一个监听器。

Example .ebextensions/httpd/conf.d/port5000.conf

```
listen 5000
<VirtualHost *:5000>
  <Proxy *>
    Require all granted
  </Proxy>
  ProxyPass / http://localhost:8080/ retry=0
  ProxyPassReverse / http://localhost:8080/
  ProxyPreserveHost on

  ErrorLog /var/log/httpd/elasticbeanstalk-error_log
</VirtualHost>
```

要完全覆盖 Elastic Beanstalk 默认 Apache 配置，请在源包的 `.ebextensions/httpd/conf/httpd.conf` 处包括一个配置。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- httpd
|       `-- conf
|           `-- httpd.conf
`-- index.jsp
```

如果要覆盖 Elastic Beanstalk Apache 配置，请将以下行添加到 `httpd.conf` 以加入适用于增强型运行状况报告和监控、响应压缩和静态文件的 Elastic Beanstalk 配置。

```
IncludeOptional conf.d/*.conf
IncludeOptional conf.d/elasticbeanstalk/*.conf
```

如果环境使用 Apache 2.2 作为其代理，则将 `IncludeOptional` 指令替换为 `Include`。有关这两个指令在两个 Apache 版本中的行为的详细信息，请参阅 Apache 2.4 中的 Include、Apache 2.4 中的 IncludeOptional 和 Apache 2.2 中的 Include。

> **Note**
>
> 要覆盖端口 80 上的默认侦听器，请在 `00_application.conf` 处包含一个名为 `.ebextensions/httpd/conf.d/elasticbeanstalk/` 的文件，以覆盖 Elastic Beanstalk 配置。

有关工作示例，请查看环境中实例上的 `/etc/httpd/conf/httpd.conf` 处的 Elastic Beanstalk 默认配置文件。您的源包中 `.ebextensions/httpd` 文件夹内的所有文件均已在部署期间被复制到 `/etc/httpd`。

扩展默认 nginx 配置 — Amazon Linux AMI (AL1)

要扩展 Elastic Beanstalk 的默认 nginx 配置，请将 `.conf` 配置文件添加到应用程序源包中名为 `.ebextensions/nginx/conf.d/` 的文件夹中。Elastic Beanstalk nginx 配置自动在此文件夹中包括 `.conf` 文件。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- nginx
|       `-- conf.d
|           |-- elasticbeanstalk
|           |   `-- my-server-conf.conf
|           `-- my-http-conf.conf
`-- index.jsp
```

`conf.d` 文件夹中扩展名为 .conf 的文件均已包括在默认配置的 `http` 数据块中。`conf.d/elasticbeanstalk` 文件夹中的文件包括在 `server` 数据块中的 `http` 数据块中。

要完全覆盖 Elastic Beanstalk 默认 nginx 配置，请在源包的 `.ebextensions/nginx/nginx.conf` 处包括一个配置。

```
~/workspace/my-app/
```

```
|-- .ebextensions
|   `-- nginx
|       `-- nginx.conf
`-- index.jsp
```

注意

- 如果要覆盖 Elastic Beanstalk nginx 配置，请将以下行添加到配置的 `server` 数据块，以加入适用于端口 80 侦听器、响应压缩和静态文件的 Elastic Beanstalk 配置。

  ```
  include conf.d/elasticbeanstalk/*.conf;
  ```

- 要覆盖端口 80 上的默认侦听器，请在 `00_application.conf` 处包含一个名为 `.ebextensions/nginx/conf.d/elasticbeanstalk/` 的文件，以覆盖 Elastic Beanstalk 配置。
- 另请将以下行包括在配置的 `http` 数据块中，以加入适用于增强型运行状况报告和监控和日志记录的 Elastic Beanstalk 配置。

  ```
      include       conf.d/*.conf;
  ```

有关工作示例，请查看环境中实例上的 `/etc/nginx/nginx.conf` 处的 Elastic Beanstalk 默认配置文件。您的源包中 `.ebextensions/nginx` 文件夹内的所有文件均已在部署期间被复制到 `/etc/nginx`。

使用 Elastic Beanstalk Java SE 平台

AWS Elastic Beanstalk Java SE 平台是一系列平台版本，用于通过已编译的 JAR 文件自行运行的 Java Web 应用程序。您可以在本地编译应用程序，或将源代码与生成脚本一起上传以便在实例中对其进行编译。Java SE 平台版本分组到平台分支中，每个分支均对应 Java 的一个主要版本，例如 Java 8 和 Java 7。

Note

Elastic Beanstalk 不会分析应用程序的 JAR 文件。将 Elastic Beanstalk 需要的文件保存在 JAR 文件外部。例如，在应用程序的源包的根目录下的 JAR 文件旁边放置工作线程环境的 `cron.yaml` 文件。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

Elastic Beanstalk Java SE 平台包含一个 nginx 服务器，该服务器用作反向代理，向您的应用程序提供缓存的静态内容并传递请求。平台提供了配置选项，以便将代理服务器配置为从源代码中的文件夹提供静态资源，从而减少应用程序上的负载。有关高级方案，您可以在源包中包括您自己的 .conf 文件，以扩展 Elastic Beanstalk 的代理配置或完全重写它。

如果您仅为应用程序源提供单个 JAR 文件（单独提供而不是包含在源包中），则 Elastic Beanstalk 会将 JAR 文件重命名为 `application.jar`，然后使用 `java -jar application.jar` 运行它。要配置在您的环境中的服务器实例上运行的进程，请在您的源包中包含一个可选的 Procfile。如果您的源包根目录中有多个 JAR 或者您要自定义 java 命令来设置 JVM 选项，必须使用 `Procfile`。

我们建议您始终在源包中将 `Procfile` 与应用程序一起提供。通过这种方式，您可以精确控制 Elastic Beanstalk 为您的应用程序运行的进程以及这些进程接收的参数。

要在部署时编译 Java 类并在您环境中的 EC2 实例上运行其他生成命令，请在应用程序源包中包含一个 Buildfile。`Buildfile` 允许您按原样部署源代码并在服务器上执行生成操作，而不是在本地编译 JAR。Java SE 平台包含常用的生成工具，允许您在服务器上执行生成操作。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅the section called “扩展 Linux 平台”。

配置 Java SE 环境

使用 Java SE 平台设置，您可以微调 Amazon EC2 实例的行为。您可以使用 Elastic Beanstalk 控制台编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，并配置应用程序可从环境中读取的变量。

在 Elastic Beanstalk 控制台中配置 Java SE 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

日志选项

“日志选项”部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

环境属性

在环境属性部分，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。环境属性会以密钥值对的形式传递到应用程序。

在运行于 Elastic Beanstalk 中的 Java SE 环境内，可通过使用 `System.getenv()` 访问环境变量。例如，您可以使用以下代码将名为 `API_ENDPOINT` 的属性读取到某个变量：

```
String endpoint = System.getenv("API_ENDPOINT");
```

参阅 环境属性和其他软件设置 了解更多信息。

Java SE 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

Java SE 平台不定义任何特定于平台的命名空间。您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间将代理配置为提供静态文件。有关详细信息和示例，请参阅the section called “静态文件”。

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

Amazon Linux AMI（在 Amazon Linux 2 之前）的 Java SE 平台

如果您的 Elastic Beanstalk Java SE 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

Java SE 配置命名空间 — Amazon Linux AMI (AL1)

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

Java SE 平台除支持所有平台支持的命名空间外，还支持一个平台特定的配置命名空间。使用 `aws:elasticbeanstalk:container:java:staticfiles` 命名空间可以定义一些选项，通过它们将您的 Web 应用程序路径映射到应用程序源包中包含静态内容的文件夹。

例如，此 option_settings 代码段在静态文件命名空间中定义了两个选项。第一个选项将路径 `/public` 映射到名为 `public` 的文件夹，第二个选项将路径 `/images` 映射到名为 `img` 的文件夹：

```
option_settings:
```

```
  aws:elasticbeanstalk:container:java:staticfiles:
    /html: statichtml
    /images: staticimages
```

通过此命名空间映射的文件夹必须为源包根目录中实际存在的文件夹。不能将路径映射到 JAR 文件中的文件夹。

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

使用 Buildfile 在服务器上生成 JAR

您可以从源包中的 `Buildfile` 文件调用生成命令，从而在环境中的 EC2 实例上生成应用程序的类文件和 JAR。

`Buildfile` 中的名只运行一次，并且在完成后必须终止，而 Procfile 文件中的命令应在应用程序的生命周期内始终运行，并且将在应用程序终止后重启。要运行应用程序中的 JAR，请使用 `Procfile`。

有关 `Buildfile` 的放置和语法的详细信息，请展开the section called “扩展 Linux 平台”中的 Buildfile 和 Procfile 部分。

以下 `Buildfile` 示例通过运行 Apache Maven 从源代码生成一个 Web 应用程序。有关使用此功能的示例应用程序，请参阅 Java Web 应用程序示例。

Example Buildfile

```
build: mvn assembly:assembly -DdescriptorId=jar-with-dependencies
```

Java SE 平台包含以下生成工具，可从您的生成脚本调用这些工具：

- `javac` – Java 编译器
- `ant` – Apache Ant
- `mvn` – Apache Maven
- `gradle` – Gradle

使用 Procfile 配置应用程序进程

如果您的应用程序源包根目录中有多个 JAR 文件，则必须包括 `Procfile` 文件来告知 Elastic Beanstalk 运行哪个或哪些 JAR。也可以为单 JAR 应用程序提供 `Procfile` 文件，以配置运行您的应用程序的 Java 虚拟机 (JVM)。

我们建议您始终在源包中将 `Procfile` 与应用程序一起提供。通过这种方式，您可以精确控制 Elastic Beanstalk 为您的应用程序运行的进程以及这些进程接收的参数。

有关编写和使用 `Procfile` 的详细信息，请展开the section called “扩展 Linux 平台”中的 Buildfile 和 Procfile 部分 。

Example Procfile

```
web: java -Xms256m -jar server.jar
cache: java -jar mycache.jar
web_foo: java -jar other.jar
```

运行应用程序中的主 JAR 的命令必须命名为 `web`，而且必须是 `Procfile` 中列出的第一条命令。nginx 服务器将它从您的环境的负载均衡器接收到的所有 HTTP 请求都转发给此应用程序。

Elastic Beanstalk 假定 Procfile 中的所有条目应始终运行，并自动重新启动 Procfile 中定义的任何终止的应用程序。要运行将会终止并且不应重新启动的命令，请使用 `Buildfile`。

在 Amazon Linux AMI（在 Amazon Linux 2 之前）上使用 Procfile

如果您的 Elastic Beanstalk Java SE 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

端口传递 — Amazon Linux AMI (AL1)

默认情况下，Elastic Beanstalk 将 nginx 代理配置为通过端口 5000 向您的应用程序转发请求。您可以覆盖默认端口，方法是将 `PORT` 环境属性设置为主应用程序侦听的端口。

如果您使用 `Procfile` 运行多个应用程序，则 Amazon Linux AMI 平台版本上的 Elastic Beanstalk 预计每个额外的应用程序都侦听编号比上一个高 100 的端口。Elastic Beanstalk 将可从每个应用程序内

访问的 PORT 变量设置为它预计该应用程序运行时使用的端口。您可以通过在应用程序代码中调用 `System.getenv("PORT")` 访问此变量。

在上面的 `Procfile` 示例中，`web` 应用程序侦听端口 5000，`cache` 侦听端口 5100，而 `web_foo` 侦听端口 5200。`web` 通过读取 `PORT` 变量配置其侦听端口，并在该端口号的基础上加 100 来确定 `cache` 侦听的端口，从而能够发送其请求。

配置反向代理

Elastic Beanstalk 使用 nginx 作为反向代理，将应用程序映射到端口 80 上的 Elastic Load Balancing 负载均衡器。Elastic Beanstalk 提供一个默认 nginx 配置，您可以扩展该配置，也可以使用您自己的配置完全覆盖该配置。

默认情况下，Elastic Beanstalk 将 nginx 代理配置为通过端口 5000 向您的应用程序转发请求。您可以覆盖默认端口，方法是将 `PORT` 环境属性设置为主应用程序侦听的端口。

> **Note**
>
> 应用程序侦听的端口不会影响 nginx 服务器为了从负载均衡器接收请求而侦听的端口。

在平台版本上配置代理服务器

所有 AL2023/AL2 平台都支持统一的代理配置功能。有关在运行 AL2023/AL2 的平台版本上配置代理服务器的更多信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

在 Amazon Linux AMI（在 Amazon Linux 2 之前）上配置代理

如果您的 Elastic Beanstalk Java SE 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的其他信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

扩展并覆盖默认代理配置 — Amazon Linux AMI（AL1）

要扩展 Elastic Beanstalk 的默认 nginx 配置，请将 `.conf` 配置文件添加到应用程序源包中名为 `.ebextensions/nginx/conf.d/` 的文件夹中。Elastic Beanstalk 的 nginx 配置自动在此文件夹中包括 `.conf` 文件。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- nginx
|       `-- conf.d
|           `-- myconf.conf
`-- web.jar
```

要完全覆盖 Elastic Beanstalk 的默认 nginx 配置，请在源包的 `.ebextensions/nginx/nginx.conf` 处包括一个配置：

```
~/workspace/my-app/
|-- .ebextensions
|   `-- nginx
|       `-- nginx.conf
`-- web.jar
```

如果要覆盖 Elastic Beanstalk 的 nginx 配置，请将以下行添加到 `nginx.conf` 中，以便加入适用于增强型运行状况报告和监控、自动应用程序映射和静态文件的 Elastic Beanstalk 配置。

```
 include conf.d/elasticbeanstalk/*.conf;
```

以下示例配置来自 Scorekeep 示例应用程序，它覆盖 Elastic Beanstalk 的默认配置，以从 `public` 的 `/var/app/current` 子目录为静态 Web 应用程序提供服务，Java SE 平台也从该处复制应用程序源代码。`/api` 位置转发流量，以在 `/api/` 下路由到侦听端口 5000 的 Spring 应用程序。所有其他流量由位于根路径的 Web 应用程序提供服务。

Example

```
user                    nginx;
error_log               /var/log/nginx/error.log warn;
pid                     /var/run/nginx.pid;
worker_processes        auto;
worker_rlimit_nofile    33282;

events {
```

```
    worker_connections  1024;
}

http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;

  log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

  include       conf.d/*.conf;

  map $http_upgrade $connection_upgrade {
      default     "upgrade";
  }

  server {
      listen        80 default_server;
      root /var/app/current/public;

      location / {
      }git pull


      location /api {
          proxy_pass          http://127.0.0.1:5000;
          proxy_http_version  1.1;

          proxy_set_header    Connection          $connection_upgrade;
          proxy_set_header    Upgrade             $http_upgrade;
          proxy_set_header    Host                $host;
          proxy_set_header    X-Real-IP           $remote_addr;
          proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
      }

      access_log    /var/log/nginx/access.log main;

      client_header_timeout 60;
      client_body_timeout   60;
      keepalive_timeout     60;
      gzip                  off;
      gzip_comp_level       4;
```

```
        # Include the Elastic Beanstalk generated locations
        include conf.d/elasticbeanstalk/01_static.conf;
        include conf.d/elasticbeanstalk/healthd.conf;
    }
}
```

向 Java 应用程序环境中添加 Amazon RDS 数据库实例

您可以使用 Amazon Relational Database Service (Amazon RDS) 数据库实例来存储应用程序收集和修改的数据。数据库可以附加到您的环境并由 Elastic Beanstalk 进行管理，也可以在外部创建和管理数据库。

如果您是首次使用 Amazon RDS，请使用 Elastic Beanstalk 控制台向测试环境中添加数据库实例，并验证您的应用程序是否可以连接到该实例。

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

添加一个数据库实例大约需要 10 分钟。环境更新完成后，您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息：

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。

属性名称	描述	属性值
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

有关配置内部数据库实例的更多信息，请参阅将数据库添加到 Elastic Beanstalk 环境。有关配置外部数据库以用于 Elastic Beanstalk 的说明，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

要连接到数据库，请将适当的驱动程序 JAR 文件添加到应用程序中，在代码中加载驱动程序类，然后使用 Elastic Beanstalk 提供的环境属性创建连接对象。

小节目录

下载 JDBC 驱动程序

您需要所选数据库引擎的 JDBC 驱动程序 JAR 文件。将 JAR 文件保存在源代码中，并在编译用于创建数据库连接的类时将该文件包含在类路径中。

在以下位置可以找到数据库引擎的最新驱动程序：

- MySQL – MySQL Connector/J
- Oracle SE-1 – Oracle JDBC Driver
- Postgres – PostgreSQL JDBC Driver
- SQL Server – Microsoft JDBC Driver

要使用 JDBC 驱动程序，请在用 `Class.forName()` 创建连接之前在代码中调用 `DriverManager.getConnection()` 加载该驱动程序。

JDBC 使用以下格式的连接字符串：

```
jdbc:driver://hostname:port/dbName?user=userName&password=password
```

您可以从 Elastic Beanstalk 提供给应用程序的环境变量中检索主机名、端口、数据库名称、用户名和密码。驱动程序名称特定于您的数据库类型和驱动程序版本。以下是驱动程序名称示例：

- `mysql` for MySQL
- `postgresql` 适用于 PostgreSQL
- `oracle:thin` 适用于 Oracle Thin
- `oracle:oci` 适用于 Oracle OCI
- `oracle:oci8` 适用于 Oracle OCI 8
- `oracle:kprb` 适用于 Oracle KPRB
- `sqlserver` 适用于 SQL Server

连接数据库 (Java SE 平台)

在 Java SE 环境中，可以使用 `System.getenv()` 从环境中读取连接变量。下面的示例代码介绍用于创建 PostgreSQL 数据库连接的类。

```
private static Connection getRemoteConnection() {
    if (System.getenv("RDS_HOSTNAME") != null) {
      try {
      Class.forName("org.postgresql.Driver");
      String dbName = System.getenv("RDS_DB_NAME");
      String userName = System.getenv("RDS_USERNAME");
      String password = System.getenv("RDS_PASSWORD");
```

```
      String hostname = System.getenv("RDS_HOSTNAME");
      String port = System.getenv("RDS_PORT");
      String jdbcUrl = "jdbc:postgresql://" + hostname + ":" + port + "/" + dbName + "?
user=" + userName + "&password=" + password;
      logger.trace("Getting remote connection with connection string from environment
 variables.");
      Connection con = DriverManager.getConnection(jdbcUrl);
      logger.info("Remote connection successful.");
      return con;
    }
    catch (ClassNotFoundException e) { logger.warn(e.toString());}
    catch (SQLException e) { logger.warn(e.toString());}
    }
    return null;
  }
```

连接数据库 (Tomcat 平台)

在 Tomcat 环境中，环境属性是以 `System.getProperty()` 可访问的系统属性的形式提供的。

下面的示例代码介绍用于创建 PostgreSQL 数据库连接的类。

```
private static Connection getRemoteConnection() {
    if (System.getProperty("RDS_HOSTNAME") != null) {
      try {
      Class.forName("org.postgresql.Driver");
      String dbName = System.getProperty("RDS_DB_NAME");
      String userName = System.getProperty("RDS_USERNAME");
      String password = System.getProperty("RDS_PASSWORD");
      String hostname = System.getProperty("RDS_HOSTNAME");
      String port = System.getProperty("RDS_PORT");
      String jdbcUrl = "jdbc:postgresql://" + hostname + ":" + port + "/" + dbName + "?
user=" + userName + "&password=" + password;
      logger.trace("Getting remote connection with connection string from environment
 variables.");
      Connection con = DriverManager.getConnection(jdbcUrl);
      logger.info("Remote connection successful.");
      return con;
    }
    catch (ClassNotFoundException e) { logger.warn(e.toString());}
    catch (SQLException e) { logger.warn(e.toString());}
    }
    return null;
```

```
  }
```

如果您无法获取连接或运行 SQL 语句，请尝试将以下代码放入 JSP 文件中。此代码连接数据库实例，创建一个表并向表中写入。

```
<%@ page import="java.sql.*" %>
<%
  // Read RDS connection information from the environment
  String dbName = System.getProperty("RDS_DB_NAME");
  String userName = System.getProperty("RDS_USERNAME");
  String password = System.getProperty("RDS_PASSWORD");
  String hostname = System.getProperty("RDS_HOSTNAME");
  String port = System.getProperty("RDS_PORT");
  String jdbcUrl = "jdbc:mysql://" + hostname + ":" +
    port + "/" + dbName + "?user=" + userName + "&password=" + password;

  // Load the JDBC driver
  try {
    System.out.println("Loading driver...");
    Class.forName("com.mysql.jdbc.Driver");
    System.out.println("Driver loaded!");
  } catch (ClassNotFoundException e) {
    throw new RuntimeException("Cannot find the driver in the classpath!", e);
  }

  Connection conn = null;
  Statement setupStatement = null;
  Statement readStatement = null;
  ResultSet resultSet = null;
  String results = "";
  int numresults = 0;
  String statement = null;

  try {
    // Create connection to RDS DB instance
    conn = DriverManager.getConnection(jdbcUrl);

    // Create a table and write two rows
    setupStatement = conn.createStatement();
    String createTable = "CREATE TABLE Beanstalk (Resource char(50));";
    String insertRow1 = "INSERT INTO Beanstalk (Resource) VALUES ('EC2 Instance');";
    String insertRow2 = "INSERT INTO Beanstalk (Resource) VALUES ('RDS Instance');";
```

```
    setupStatement.addBatch(createTable);
    setupStatement.addBatch(insertRow1);
    setupStatement.addBatch(insertRow2);
    setupStatement.executeBatch();
    setupStatement.close();

  } catch (SQLException ex) {
    // Handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
  } finally {
    System.out.println("Closing the connection.");
    if (conn != null) try { conn.close(); } catch (SQLException ignore) {}
  }

  try {
    conn = DriverManager.getConnection(jdbcUrl);

    readStatement = conn.createStatement();
    resultSet = readStatement.executeQuery("SELECT Resource FROM Beanstalk;");

    resultSet.first();
    results = resultSet.getString("Resource");
    resultSet.next();
    results += ", " + resultSet.getString("Resource");

    resultSet.close();
    readStatement.close();
    conn.close();

  } catch (SQLException ex) {
    // Handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
  } finally {
       System.out.println("Closing the connection.");
      if (conn != null) try { conn.close(); } catch (SQLException ignore) {}
  }
%>
```

要显示结果，请将以下代码放入 JSP 文件 HTML 部分的正文中。

```
<p>Established connection to RDS. Read first two rows: <%= results %></p>
```

数据库连接问题排查

如果您从应用程序中连接数据库时遇到问题，请查看 Web 容器日志和数据库。

查看日志

您可以从 Eclipse 中查看 Elastic Beanstalk 环境中的所有日志。如果未打开 AWS Explorer 视图，请选择工具栏中的橙色 AWS 图标旁边的箭头，然后选择 Show AWS Explorer View（显示 Amazon Explorer 视图）。展开 AWS Elastic Beanstalk 和您的环境名称，然后打开服务器的上下文（右键单击）菜单。选择 Open in WTP Server Editor（在 WTP Server Editor 中打开）。

选择 Server（服务器）视图的 Log（日志）选项卡以查看环境中的聚合日志。要打开最新日志，请选择页面右上角的 Refresh（刷新）按钮。

向下滚动以在 `/var/log/tomcat7/catalina.out` 中找到 Tomcat 日志。如果您已在前面的示例中多次加载网页，可能会看到以下内容。

```
-------------------------------------
/var/log/tomcat7/catalina.out
-------------------------------------
INFO: Server startup in 9285 ms
Loading driver...
Driver loaded!
SQLException: Table 'Beanstalk' already exists
SQLState: 42S01
VendorError: 1050
Closing the connection.
Closing the connection.
```

Web 应用程序发送到标准输出的所有信息都会显示在 Web 容器日志中。在上一示例中，每次加载页面时，应用程序都会尝试创建表。这导致在第一次页面加载后，每次页面加载时都会捕获 SQL 异常。

作为示例，上述行为是可接受的。但在实际应用程序中，应将数据库定义保留在架构对象中，从模型类中执行事务并与控制器 servlet 协调请求。

连接到 RDS 数据库实例

您可以使用 MySQL 客户端应用程序直接连接到 Elastic Beanstalk 环境中的 RDS 数据库实例。

您首先需要对 RDS 数据库实例开放安全组以允许来自计算机的流量。

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database (数据库) 配置类别中，选择 Edit (编辑)。
5. 在 Endpoint（端点）旁边，选择 Amazon RDS 控制台链接。
6. 在 RDS Dashboard（RDS 控制面板）实例详细信息页面上的 Security and Network（安全与网络）下，选择 Security Groups（安全组）旁边的以 rds- 开头的安全组。

> **Note**
>
> 数据库可能有多个标记为 Security Groups（安全组）的条目。仅当您拥有不具有默认 Amazon Virtual Private Cloud (Amazon VPC) 的旧账户时，才应使用第一个条目（以 awseb 开头）。

7. 在 Security group details（安全组详细信息）中，选择 Inbound（入站）选项卡，然后选择 Edit（编辑）。
8. 为 MySQL (端口 3306) 添加一个允许来自 IP 地址 (以 CIDR 格式指定) 的流量的规则。
9. 选择 Save。更改将立即生效。

返回到环境的 Elastic Beanstalk 配置详细信息并记下终端节点。您将使用域名连接到 RDS 数据库实例。

安装 MySQL 客户端并在端口 3306 上发起与数据库的连接。在 Windows 上，从 MySQL 主页安装 MySQL Workbench 并按照提示操作。

在 Linux 上，使用适用于您的分发版的程序包管理器安装 MySQL 客户端。以下示例对 Ubuntu 和其他 Debian 衍生物有效。

```
// Install MySQL client
$ sudo apt-get install mysql-client-5.5
...
```

```
// Connect to database
$ mysql -h aas839jo2vwhwb.cnubrrfwfka8.us-west-2.rds.amazonaws.com -u username -
ppassword ebdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 117
Server version: 5.5.40-log Source distribution
...
```

连接之后，您可以运行 SQL 命令以查看数据库状态，了解是否已创建表和行及其他信息。

```
mysql> SELECT Resource from Beanstalk;
+--------------+
| Resource     |
+--------------+
| EC2 Instance |
| RDS Instance |
+--------------+
2 rows in set (0.01 sec)
```

使用 AWS Toolkit for Eclipse

AWS Toolkit for Eclipse 将 AWS Elastic Beanstalk 管理功能与您的 Tomcat 开发环境集成，以帮助进行环境创建、配置和代码部署。该工具包可支持多个 AWS 账户、管理现有环境以及直接连接到环境中的实例以进行故障排除。

> **Note**
>
> AWS Toolkit for Eclipse 仅支持将 Java 与 Tomcat 平台（而不是 Java SE 平台）结合使用的项目。

有关先决条件和安装 AWS Toolkit for Eclipse 的更多信息，请转到 https://aws.amazon.com/eclipse。您还可以查看将 AWS Elastic Beanstalk 与 AWS Toolkit for Eclipse 配合使用视频。本主题还介绍了下列各种有用信息，包括工具、介绍基础知识的主题和其他资源，供 Java 开发人员使用。

将现有的环境导入 Eclipse

您可以将在 AWS 管理控制台中创建的现有环境导入到 Eclipse 中。

要导入现有环境，请展开 AWS Elastic Beanstalk 节点，并双击 Eclipse 内的 AWS Explorer 中的环境。现在，您可以将 Elastic Beanstalk 应用程序部署到此环境中。

管理 Elastic Beanstalk 应用程序环境

主题

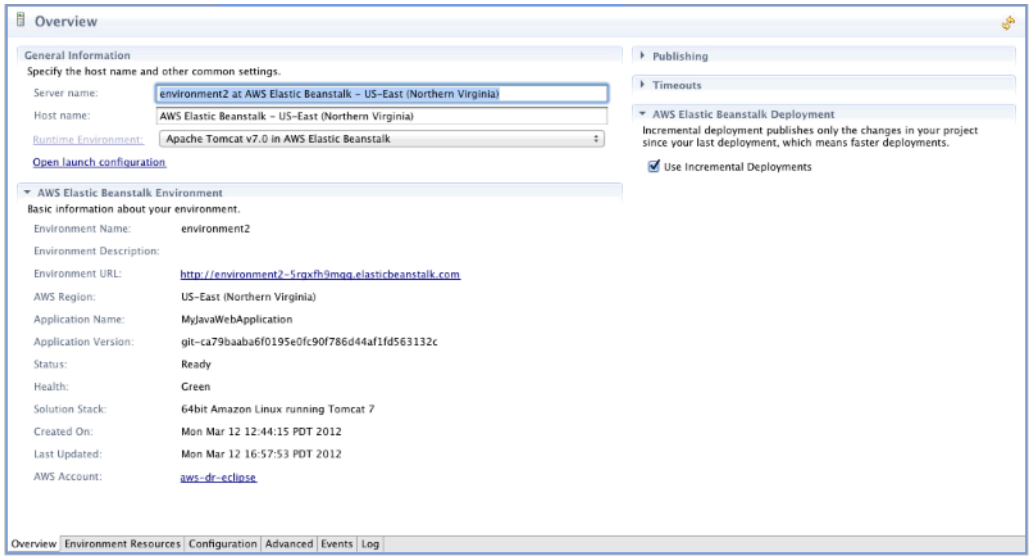
借助 AWS Toolkit for Eclipse，可以更改应用程序环境使用的 AWS 资源的预置和配置。有关如何使用 AWS 管理控制台管理应用程序环境的详细信息，请参阅管理环境。此部分讨论在应用程序环境配置过程中可以在 AWS Toolkit for Eclipse 中编辑的特定服务设置。有关 AWS Toolkit for Eclipse 的更多信息，请参阅 AWS Toolkit for Eclipse 入门指南。

更改环境配置设置

在部署应用程序时，Elastic Beanstalk 会配置许多 AWS 云计算服务。您可以使用 AWS Toolkit for Eclipse 控制如何配置这些个别服务。

编辑应用程序的环境设置

1. 如果 Eclipse 不显示 AWS Explorer 视图，请在菜单中依次选择 Window（窗口）、Show View（显示视图）和 AWS Explorer。展开 Elastic Beanstalk 节点和应用程序节点。
2. 在 AWS Explorer 中，双击您的 Elastic Beanstalk 环境。
3. 在窗格底部，单击 Configuration（配置）选项卡。

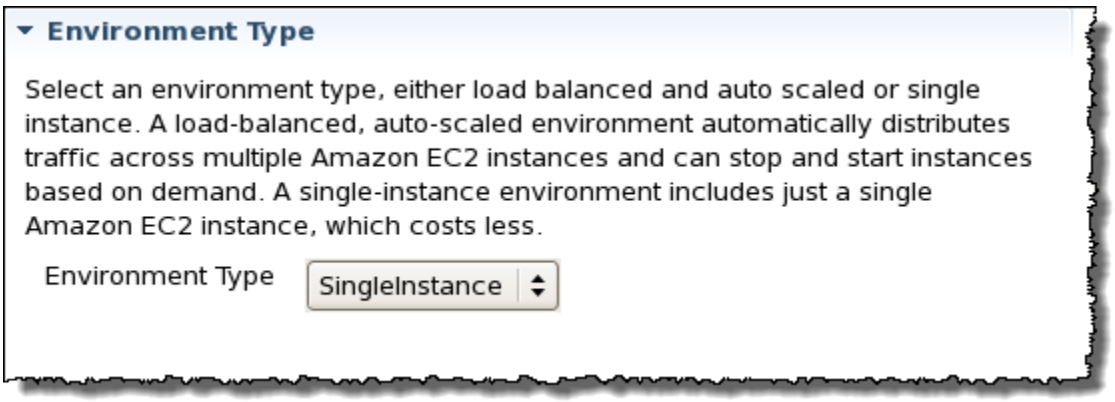


您现在可以配置以下各项的设置：

- EC2 服务器实例
- 负载均衡器
- AutoScaling
- 通知
- 环境类型
- 环境属性

更改环境类型

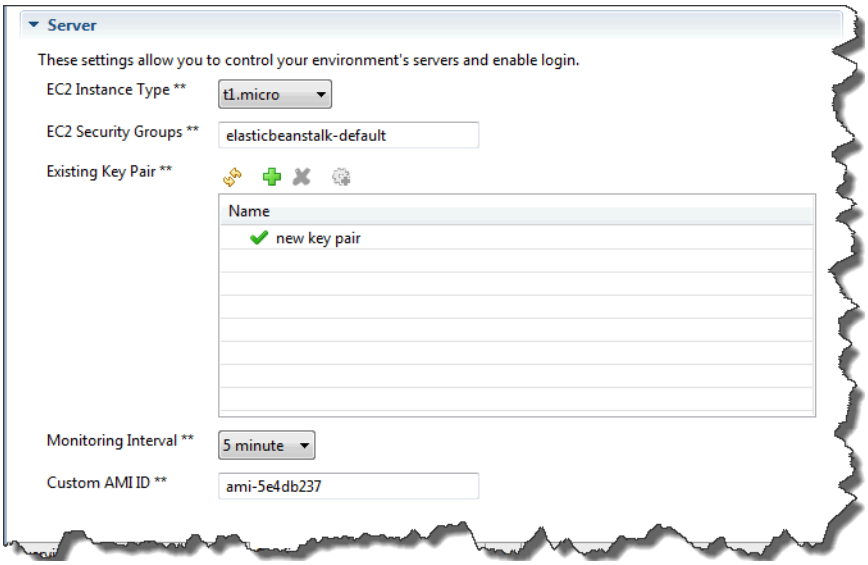
在 AWS Toolkit for Eclipse 中，通过您环境的 Configuration（配置）选项卡的 Environment Type（环境类型）部分，您可以选择 Load balanced, auto scaled（负载均衡、自动扩展）或 Single instance（单一实例）环境，具体取决于您部署的应用程序的要求。对于需要可扩展性的应用程序，选择 Load balanced, auto scaled（负载均衡、自动扩展）。对于简单的低流量应用程序，选择 Single instance（单一实例）。有关更多信息，请参阅环境类型。



使用 配置 EC2 服务器实例AWS Toolkit for Eclipse

Amazon Elastic Compute Cloud (EC2) 是一项 Web 服务，用于在 Amazon 数据中心启动和管理服务器实例。您可以随时使用 Amazon EC2 服务器实例，并可根据需要在任意长的时间内使用，也可以用于任何合法用途。实例可以按照不同的规模和配置进行提供。有关更多信息，请转到 Amazon EC2 产品页面。

在 Toolkit for Eclipse 内的环境 Configuration（配置）选项卡上的 Server（服务器）下，您可以编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。



Amazon EC2 实例类型

Instance type（实例类型）显示可用于您的 Elastic Beanstalk 应用程序的实例类型。请更改实例类型以选择特征（包括内存大小和 CPU 处理能力）最适合您的应用程序的服务器。例如，具有大量操作和长时间运行的操作的应用程序可能需要更多 CPU 或内存。

有关可用于 Elastic Beanstalk 应用程序的 Amazon EC2 实例类型的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的实例类型。

Amazon EC2 安全组

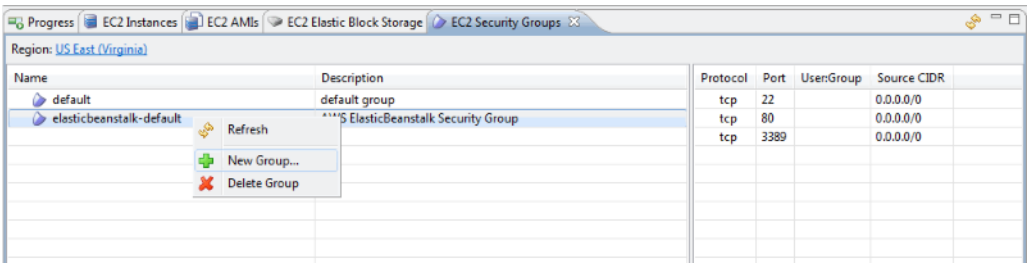
您可以使用 Amazon EC2 安全组 控制对 Elastic Beanstalk 应用程序的访问。安全组会定义实例的防火墙规则。这些规则会指定应将哪些进入 (即传入) 网络流量提交给实例。将丢弃所有其他进入流量。您可以随时针对不同的组修改这些规则。新的规则会自动在所有现在运行的和将来启动的实例上强制实施。

您可以使用 AWS 管理控制台或 AWS Toolkit for Eclipse 设置 Amazon EC2 安全组。通过在 EC2 Security Groups（EC2 安全组）框中输入一个或多个 Amazon EC2 安全组名称（用逗号分隔），您可以指定哪些 Amazon EC2 安全组控制对您的 Elastic Beanstalk 应用程序的访问。

> **Note**
>
> 如果您正在运行使用早期容器类型的应用程序，且想为应用程序启用运行状况检查，请确保可从 0.0.0.0/0 的源 CIDR 范围访问端口 80 (HTTP)。有关运行状况检查的详细信息，请参阅 运行状况检查。要检查您使用的是否是早期容器类型，请参阅the section called “为什么某些平台版本标记为传统版本？”。

使用 AWS Toolkit for Eclipse 创建安全组

1. 在 AWS Toolkit for Eclipse 中，单击 AWS Explorer 选项卡。展开 Amazon EC2 节点，然后双击 Security Groups（安全组）。
2. 右键单击左表中的任意位置，然后单击 New Group（新建组）。



3. 在 Security Group（安全组）框中，键入安全组的名称和描述，然后单击 OK（确定）。

有关 Amazon EC2 安全组的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的使用安全组。

Amazon EC2 密钥对

您可以使用 Amazon EC2 密钥对安全地登录到为 Elastic Beanstalk 应用程序预配置的 Amazon EC2 实例。

> **Important**
>
> 您必须创建 Amazon EC2 密钥对并将 Elastic Beanstalk 预配置的 Amazon EC2 实例配置为使用 Amazon EC2 密钥对，然后才能访问 Elastic Beanstalk 预配置的 Amazon EC2 实例。在向 Elastic Beanstalk 部署应用程序时，您可以使用 AWS Toolkit for Eclipse 内的 Publish to Beanstalk Wizard（发布到 Beanstalk 向导）创建密钥对。也可以使用 AWS 管理控制台设置 Amazon EC2 密钥对。有关为 Amazon EC2 创建密钥对的说明，请参阅 Amazon Elastic Compute Cloud 入门指南。

有关 Amazon EC2 密钥对的更多信息，请转到 Amazon Elastic Compute Cloud 用户指南 中的使用 Amazon EC2 凭证。有关连接到 Amazon EC2 实例的更多信息，请转到 Amazon Elastic Compute Cloud 用户指南 中的连接到实例和使用 PuTTY 从 Windows 连接到 Linux/UNIX 实例。

CloudWatch 指标

默认情况下，仅启用基本 Amazon CloudWatch 指标。这些指标会以五分钟为周期返回数据。在 AWS Toolkit for Eclipse 中，您可为环境启用更精细的一分钟 CloudWatch 指标，方法为在环境的 Configuration（配置）选项卡的 Server（服务器）部分中为 Monitoring Interval（监控间隔）选择 1 minute（1 分钟）。

> **Note**
>
> 一分钟时间间隔指标可能产生 Amazon CloudWatch 服务费用。有关更多信息，请参阅 Amazon CloudWatch。

自定义 AMI ID

在 AWS Toolkit for Eclipse 中，您可将自定义 AMI 的标识符输入到环境的 Configuration（配置）选项卡的 Server（服务器）部分中的 Custom AMI ID（自定义 AMI ID）框，以便使用您自己的自定义 AMI 覆盖用于 Amazon EC2 实例的默认 AMI。

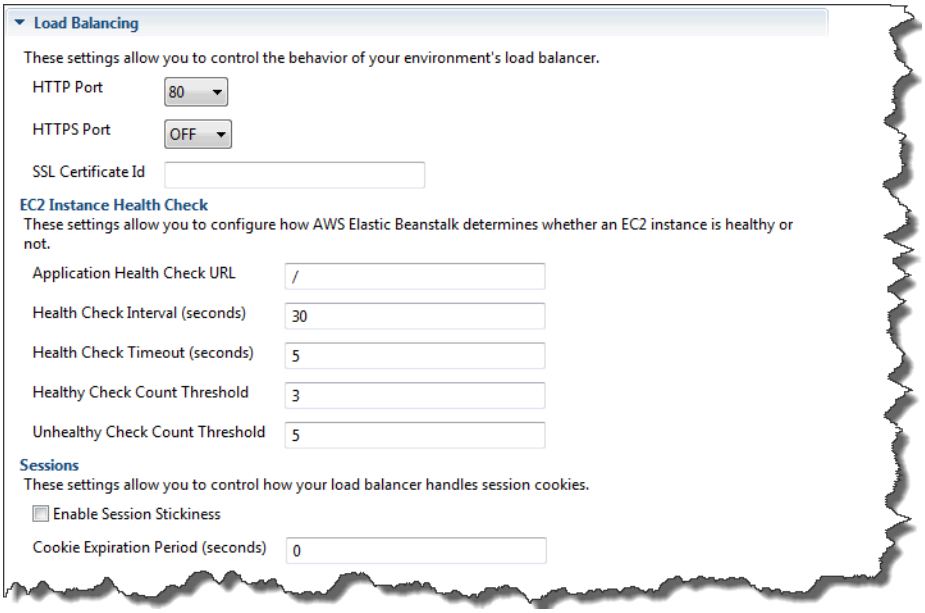
> **Important**
>
> 使用您自己的 AMI 是一项高级任务，应小心谨慎地执行。如果需要自定义 AMI，则建议您先使用默认 Elastic Beanstalk AMI，然后修改它。若要保持正常运行状态，Elastic Beanstalk 期望 Amazon EC2 实例满足一系列要求，包括具有一个正在运行的主机管理器。如果未满足这些要求，您的环境可能无法正常运行。

使用 配置 Elastic Load BalancingAWS Toolkit for Eclipse

Elastic Load Balancing 是一种 Amazon Web 服务，可帮助您提高应用程序的可用性和可扩展性。Elastic Load Balancing 可让您在两个或更多的 Amazon EC2 实例之间分配应用程序负载。Elastic Load Balancing 通过冗余提高可用性，并支持应用程序的流量增长。

Elastic Load Balancing 可让您自动在运行的所有 EC2 服务器实例之间分配和平衡传入的应用程序流量。在您需要增加应用程序容量时，该服务还可让您轻松地添加新的实例。

Elastic Beanstalk 会在您部署应用程序时自动地预配置 Elastic Load Balancing。在 Toolkit for Eclipse 内的环境 Configuration（配置）选项卡上的 Load Balancing（负载均衡）下，您可以编辑 Elastic Beanstalk 环境的负载均衡配置。



以下部分介绍了可为应用程序配置的 Elastic Load Balancing 参数。

端口

预配置来处理您的 Elastic Beanstalk 应用程序请求的负载均衡器会将请求发送到正在运行您的应用程序的 Amazon EC2 实例。预配置的负载均衡器会侦听 HTTP 和 HTTPS 端口上的请求，并将请求路由到 AWS Elastic Beanstalk 应用程序中的 Amazon EC2 实例。默认情况下，负载均衡器将处理 HTTP 端口上的请求。必须至少打开其中一个端口，要么是 HTTP 要么是 HTTPS。

Important

确保您指定的端口没有锁定；否则，用户将无法连接到 Elastic Beanstalk 应用程序。

控制 HTTP 端口

要关闭 HTTP 端口，请为 HTTP Listener Port（HTTP 侦听器端口）选择 OFF。若要打开 HTTP 端口，请选择一个 HTTP 端口（例如 80）。

Note

要使用默认端口 80 (如端口 8080) 以外的端口来访问您的环境，请将侦听器添加到现有负载均衡器并配置新侦听器来侦听该端口。
例如，当使用适用于 Classic Load Balancer 的 AWS CLI 时，键入以下命令可将 *LOAD_BALANCER_NAME* 替换为您用于 Elastic Beanstalk 的负载均衡器的名称。

```
aws elb create-load-balancer-listeners --load-balancer-name LOAD_BALANCER_NAME
 --listeners "Protocol=HTTP, LoadBalancerPort=8080, InstanceProtocol=HTTP,
 InstancePort=80"
```

例如，当使用适用于 Application Load Balancer 的 AWS CLI 时，键入以下命令可将 *LOAD_BALANCER_ARN* 替换为您用于 Elastic Beanstalk 的负载均衡器的 ARN。

```
aws elbv2 create-listener --load-balancer-arn LOAD_BALANCER_ARN --protocol HTTP
 --port 8080
```

如果希望 Elastic Beanstalk 监控您的环境，请勿删除端口 80 上的侦听器。

控制 HTTPS 端口

Elastic Load Balancing 支持 HTTPS/TLS 协议，可为负载均衡器的客户端连接流量加密。从负载均衡器到 EC2 实例的连接是使用纯文本完成的。默认情况下，HTTPS 端口是关闭的。

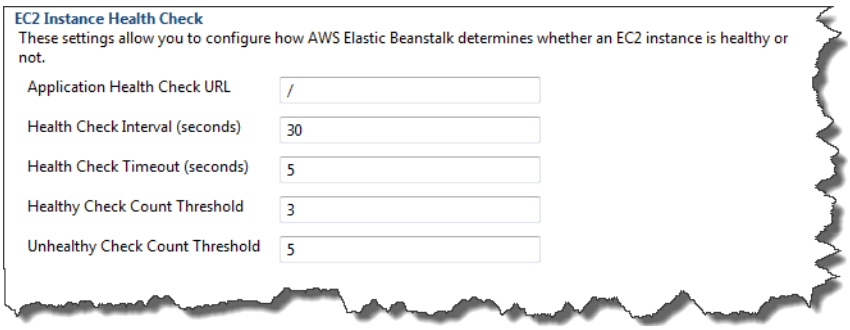
打开 HTTPS 端口

1. 使用 AWS Certificate Manager (ACM) 创建新的证书，或者将证书和密钥上传到 AWS Identity and Access Management (IAM)。有关请求 ACM 证书的更多信息，请参阅 AWS Certificate Manager 用户指南中的请求证书。有关将第三方证书导入 ACM 中的更多信息，请参阅 AWS Certificate Manager 用户指南中的导入证书。如果 ACM 在您所在的 AWS 区域不可用，请使用 AWS Identity and Access Management (IAM) 上传第三方证书。ACM 和 IAM 服务存储证书并为 SSL 证书提供 Amazon Resource Name (ARN)。有关创建证书并将证书上传到 IAM 的更多信息，请参阅 IAM 用户指南 中的使用服务器证书。
2. 在 HTTPS Listener Port（HTTPS 侦听器端口）下拉列表中选择一个端口，将其指定为 HTTPS 端口。
3. 在 SSL Certificate ID（SSL 证书 ID）文本框中，输入 SSL 证书的 Amazon Resource Name (ARN)。例如，**`arn:aws:iam::123456789012:server-certificate/abc/certs/build`** 或 **`arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`**。使用您在步骤 1 中创建和上传的 SSL 证书。

若要关闭 HTTPS 端口，请为 HTTPS Listener Port (HTTPS 侦听器端口) 选择 OFF (关)。

运行状况检查

您可以使用 Load Balancing (负载均衡) 面板的 EC2 Instance Health Check (EC2 实例运行状况检查) 部分来控制运行状况检查的设置。



下表介绍了可为您的应用程序设置的运行状况检查参数。

- 为了确定实例运行状况，Elastic Beanstalk 会在其查询的 URL 上查找 200 响应代码。默认情况下，Elastic Beanstalk 对非早期容器检查 TCP:80，对早期容器检查 HTTP:80。您可以通过在 Application Health Check URL（应用程序运行状况检查 URL）框中输入 URL（例如 **`/myapp/index.jsp`**）以覆盖默认 URL，使之对应于您的应用程序中的现有资源。如果您覆盖默认 URL，Elastic Beanstalk 将使用 HTTP 查询资源。要检查您使用的是否是早期容器类型，请参阅the section called “为什么某些平台版本标记为传统版本？”。
- 对于 Health Check Interval (seconds)（运行状况检查间隔 (秒)），输入应用程序 Amazon EC2 实例运行状况检查之间的秒数。
- 对于 Health Check Timeout（运行状况检查超时），指定 Elastic Load Balancing 在将实例视为无响应之前等待响应的秒数。
- 使用 Healthy Check Count Threshold（良好运行状况检查计数阈值）和 Unhealthy Check Count Threshold（不佳运行状况检查计数阈值）框，指定 Elastic Load Balancing 更改实例的运行状况状态之前连续的成功或失败 URL 探测的次数。例如，在 Unhealthy Check Count Threshold（不佳运行状况检查计数阈值）文本框中填入 5，即表示必须在该 URL 连续返回 5 次错误消息或者超时后，Elastic Load Balancing 才会将运行状况检查确定为“失败”。

会话

默认情况下，负载均衡器会以最小的负载将每个请求独立地传送给该服务器实例。比较起来，粘性 (VPC) 会将用户的会话绑定到具体的服务器实例，以便该用户在会话期间发出的所有请求都会发送到同一个服务器实例中。

在为应用程序启用粘性会话后，Elastic Beanstalk 会使用负载均衡器生成的 HTTP Cookie。负载均衡器会使用负载均衡器生成的特别 Cookie 来跟踪每个请求的应用程序实例。在负载均衡器收到请求时，它首先会检查并查看请求中是否存在这个 Cookie。如果是这样的话，该请求会发送到 Cookie 中指定的应用程序实例。如果负载均衡器找不到 Cookie，它会根据现有的负载均衡算法选择一个应用程序实

例。响应中会插入 Cookie，从而将同一用户发出的后续请求绑定到该应用程序实例中。策略配置会定义 Cookie 的到期时间，从而确定每个 Cookie 的有效持续时间。

在 Load Balancer（负载均衡器）下的 Sessions（会话）部分，指定是否让应用程序的负载均衡器支持会话粘性和每个 Cookie 的持续时间。

有关 Elastic Load Balancing 的更多信息，请转到 Elastic Load Balancing 开发人员指南。

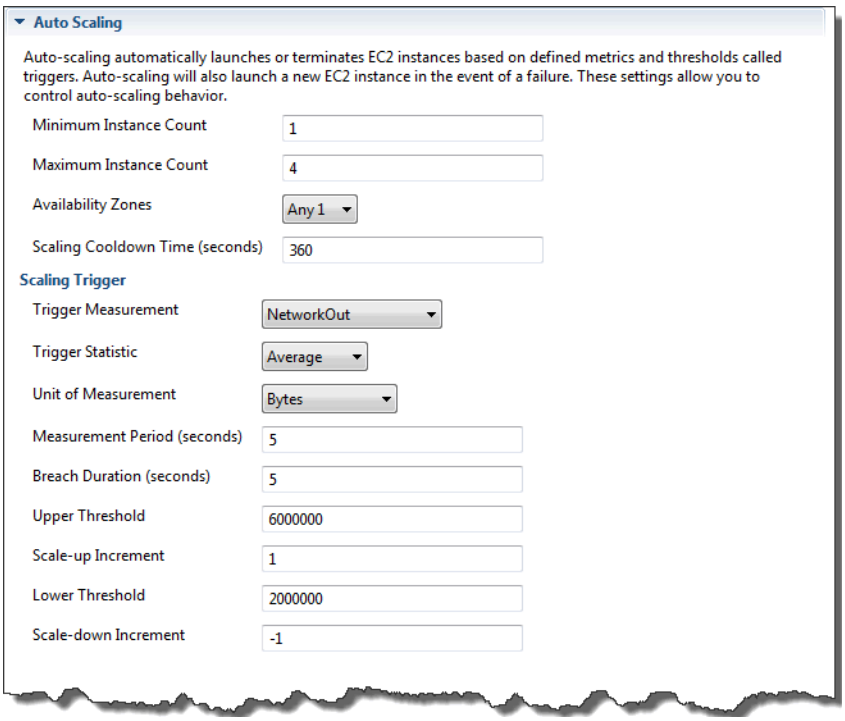
使用 配置 Auto ScalingAWS Toolkit for Eclipse

Amazon EC2 Auto Scaling 是一项 Amazon Web 服务，旨在根据用户定义的触发器自动启动或终止 Amazon EC2 实例。用户可以设置 Auto Scaling 组 并将 触发器 与这些组关联，以根据带宽使用量或 CPU 利用率等指标自动扩展计算资源。Amazon EC2 Auto Scaling 与 Amazon CloudWatch 协作，检索运行应用程序的服务器实例的指标。

借助 Amazon EC2 Auto Scaling，您可以获取一组 Amazon EC2 实例并设置各种参数，使此组的数量自动增加或减少。Amazon EC2 Auto Scaling 可以在该组中添加或删除 Amazon EC2 实例，以帮助您无缝处理应用程序的流量变化。

Amazon EC2 Auto Scaling 还会监控其启动的每个 Amazon EC2 实例的运行状况。如果任何实例意外终止，Amazon EC2 Auto Scaling 会检测到终止情况并启动替换实例。这一功能可让您自动维护固定的、预期数量的 Amazon EC2 实例。

Elastic Beanstalk 为您的应用程序预配置 Amazon EC2 Auto Scaling。在 Toolkit for Eclipse 内的环境 Configuration（配置）选项卡上的 Auto Scaling 下，您可以编辑 Elastic Beanstalk 环境的 Auto Scaling 配置。



以下部分介绍了如何为您的应用程序配置 Auto Scaling 参数。

启动配置

您可以编辑启动配置以控制 Elastic Beanstalk 应用程序如何预配置 Amazon EC2 Auto Scaling 资源。

使用 Minimum Instance Count（最小实例计数）和 Maximum Instance Count（最大实例计数）设置，指定 Elastic Beanstalk 应用程序使用的 Auto Scaling 组的最小大小和最大大小。

> **Note**
>
> 要保持固定数量的 Amazon EC2 实例，请将 Minimum Instance Count（最小实例计数）和 Maximum Instance Count（最大实例计数）文本框设置为相同的值。

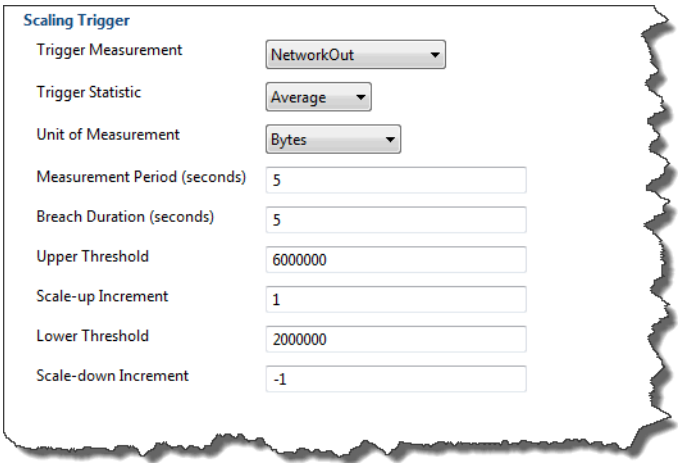
对于 Availability Zones（可用区），指定您希望 Amazon EC2 实例加入的可用区的数量。如果要构建具有容错能力的应用程序，则设置此数量很重要：如果一个可用区出现故障，您的实例仍将在其他可用区中运行。

> **Note**
>
> 目前，您无法指定您的实例将放入哪些可用区域。

触发

触发器 是您设置的 Amazon EC2 Auto Scaling 机制，用于告知系统何时增加（扩展）和减少（缩减）实例数量。您可以配置触发器，使其在将任何指标（如 CPU 利用率）发布到 Amazon CloudWatch 时激发，并确定是否已满足指定的条件。当超过该指标的上限或下限达到指定的时间长度时，该触发器会启动一个名为扩展活动 的长时间运行的进程。

您可以使用 AWS Toolkit for Eclipse 为 Elastic Beanstalk 应用程序定义扩展触发器。



您可以在 Toolkit for Eclipse 内您环境的 Configuration（配置）选项卡的 Scaling Trigger（扩展触发器）部分中，配置以下触发器参数列表。

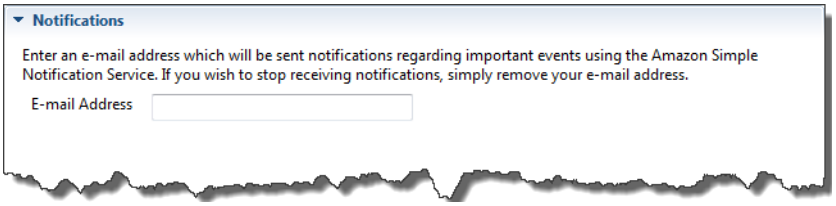
- 对于 Trigger Measurement（触发器管理），指定触发器的指标。
- 对于 Trigger Statistic（触发器统计数据），指定触发器将使用的统计数据 – **`Minimum`**、**`Maximum`**、**`Sum`** 或 **`Average`**。
- 对于 Unit of Measurement（测量单位），指定触发器度量单位。
- 对于 Measurement Period（测量周期），指定 Amazon CloudWatch 对触发器的指标进行度量的频率。对于 Breach Duration（违例持续时间），指定在激活触发器之前，指标可以超出所定义限制范围 [如 Upper Threshold（上限）和 Lower Threshold（下限）所指定] 的时长。

- 对于 Scale-up Increment（扩展增量）和 Scale-down Increment（缩减增量），指定在执行扩展活动时要添加或删除的 Amazon EC2 实例数。

有关 Amazon EC2 Auto Scaling 的更多信息，请参阅 Amazon Elastic Compute Cloud 文档上的 Amazon EC2 Auto Scaling 部分。

使用 配置通知AWS Toolkit for Eclipse

Elastic Beanstalk 使用 Amazon Simple Notification Service (Amazon SNS) 向您通知影响应用程序的重要事件。要启用 Amazon SNS 通知，只需在 Toolkit for Eclipse 内，在环境 Configuration（配置）选项卡上的 Notifications（通知）下的 Email Address（电子邮件地址）文本框中，输入您的电子邮件地址。要禁用 Amazon SNS 通知，可从文本框中删除电子邮件地址。



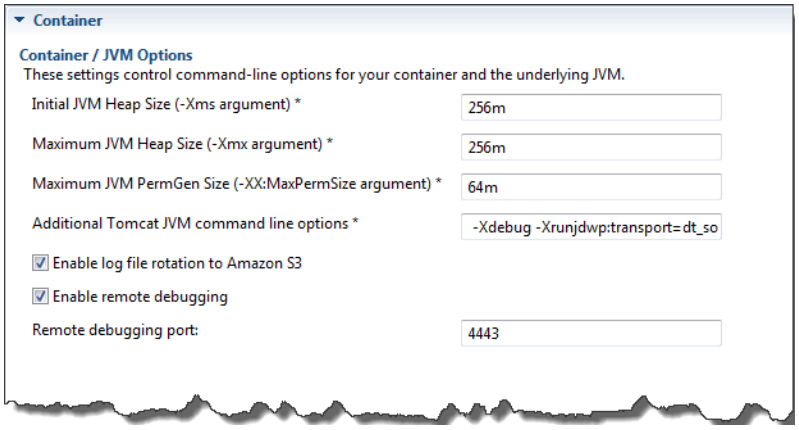
使用 配置 Java 容器AWS Toolkit for Eclipse

通过 Container/JVM Options（容器/JVM 选项）面板，您可以微调 Java 虚拟机在 Amazon EC2 实例上的行为，并启用或禁用 Amazon S3 日志轮换。您可以使用 AWS Toolkit for Eclipse 配置您的容器信息。有关可用于 Tomcat 环境的选项的更多信息，请参阅 the section called “配置 Tomcat 环境”。

> **Note**
>
> 您可以通过切换环境别名记录的方法将配置设置修改为零故障时间。有关更多信息，请参阅使用 Elastic Beanstalk 进行蓝/绿部署。

访问 Elastic Beanstalk 应用程序的“容器/JVM 选项”面板

1. 如果 Eclipse 不显示 AWS Explorer 视图，请在菜单中依次选择 Window (窗口)、Show View (显示视图) 和 AWS Explorer。展开 Elastic Beanstalk 节点和应用程序节点。
2. 在 AWS Explorer 中，双击您的 Elastic Beanstalk 环境。
3. 在窗格底部，单击 Configuration (配置) 选项卡。
4. 在 Container (容器) 下，您可以配置容器的选项。



远程调试

为了远程测试您的应用程序，您可以在调试模式中运行应用程序。

启用远程调试

1. 选择 Enable remote debugging（启用远程调试）。
2. 对于 Remote debugging port（远程调试端口），指定用于远程调试的端口号。

   Additional Tomcat JVM command line options（更多 Tomcat JVM 命令行选项）设置将自动填充。

启动远程调试

1. 在 AWS Toolkit for Eclipse 菜单中，依次选择 Window（窗口）、Show View（显示视图）和 Other（其他）。
2. 展开 Server（服务器）文件夹，然后选择 Servers（服务器）。选择 OK。
3. 在 Servers（服务器）窗格中，右键单击您的应用程序正在其中运行的服务器，然后单击 Restart in Debug（在调试中重新启动）。

通过 AWS Toolkit for Eclipse 设置系统属性

以下示例在 AWS Toolkit for Eclipse 中设置 `JDBC_CONNECTION_STRING` 系统属性。设置此属性之后，它可作为一个名为 `JDBC_CONNECTION_STRING` 的系统属性供 Elastic Beanstalk 应用程序使用。

> **Note**
>
> AWS Toolkit for Eclipse 尚不支持在 VPC 中修改环境的环境配置，包括系统属性。除非您拥有使用 EC2 Classic 的旧账户，否则必须使用 AWS 管理控制台（下一部分中将进行说明）或 EB CLI。

> **Note**
>
> 环境配置设置可以包含任何可打印的 ASCII 字符 (重读符号“`”除外，即 ASCII 96)，且长度不能超过 200 个字符。

为 Elastic Beanstalk 应用程序设置系统属性

1. 如果 Eclipse 不显示 AWS Explorer 视图，请依次选择 Window（窗口）、Show View（显示视图）、Other（其他）。展开 AWS Toolkit，然后选择 AWS Explorer。
2. 在 AWS Explorer 窗格中，展开 Elastic Beanstalk，展开您的应用程序的节点，然后双击您的 Elastic Beanstalk 环境。
3. 在该环境的窗格底部，单击 Advanced（高级）选项卡。
4. 在 aws:elasticbeanstalk:application:environment 下面，单击 JDBC_CONNECTION_STRING，然后键入连接字符串。例如，下面的 JDBC 连接字符串将使用用户名 `me` 和密码 `mypassword` 连接到本地主机端口 3306 上的 MySQL 数据库实例：

   ```
   jdbc:mysql://localhost:3306/mydatabase?user=me&password=mypassword
   ```

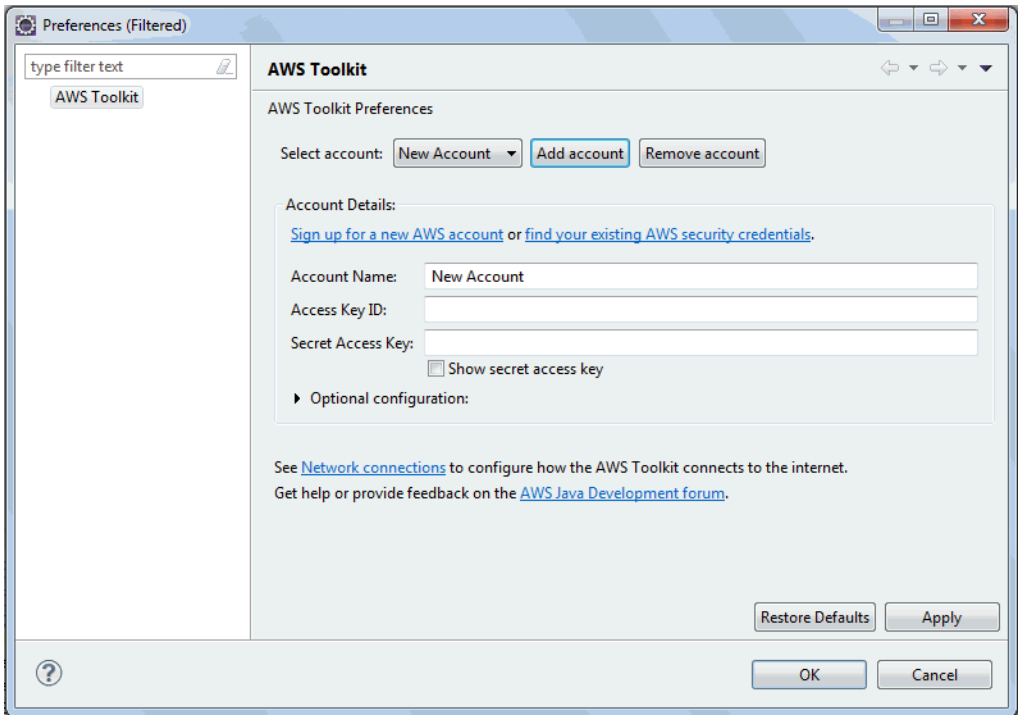
   您的 Elastic Beanstalk 应用程序可通过名为 `JDBC_CONNECTION_STRING` 的系统属性访问该字符串。
5. 在键盘上按 Ctrl+C 或者依次选择 File（文件）和 Save（保存）来保存对环境配置所做的更改。这些更改将在约 1 分钟内反映出来。

管理多个 AWS 账户

建议您为执行不同的任务（如测试、暂存和生产）而设置不同的 AWS 账户。您可以使用 AWS Toolkit for Eclipse 轻松地添加、编辑和删除账户。

使用 AWS Toolkit for Eclipse 添加 AWS 账户

1. 在 Eclipse 中，确保工具栏可见。在工具栏上，单击 AWS 图标旁的箭头，然后选择 Preferences（首选项）。
2. 单击 Add account（添加账户）。



3. 在 Account Name（账户名称）文本框中，键入账户的显示名称。
4. 在 Access Key ID（访问密钥 ID）文本框中，键入 AWS 访问密钥 ID。
5. 在 Secret Access Key（秘密访问密钥）文本框中，键入 AWS 私有密钥。

   对于 API 访问，您需要访问密钥 ID 和秘密访问密钥。使用 IAM 用户访问密钥而不是 AWS 账户根用户 访问密钥。有关创建访问密钥的更多信息，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。
6. 单击 OK (确定)。

使用不同的账户将应用程序部署到 Elastic Beanstalk

1. 在 Eclipse 工具栏中，单 AWS 击图标旁的箭头，然后选择 Preferences（首选项）。
2. 对于 Default Account（默认账户），选择要用于将应用程序部署到 Elastic Beanstalk 的账户。
3. 单击 OK (确定)。

4. 在 Project Explorer（项目资源管理器）窗格中，右键单击要部署的应用程序，然后选择 Amazon Web Services > Deploy to Elastic Beanstalk（部署到 Elastic Beanstalk）。

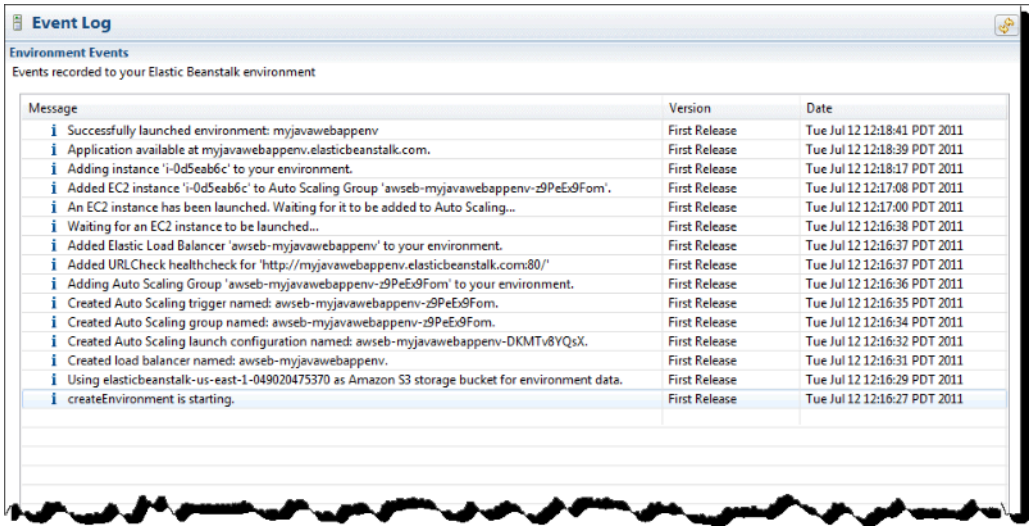
查看 事件

您可以使用 AWS Toolkit for Eclipse 访问与该应用程序相关联的活动和通知。

查看应用程序事件

1. 如果 Eclipse 不显示 AWS Explorer 视图，请在菜单中单击 Window（窗口）> Show View（显示视图）> AWS Explorer。展开 Elastic Beanstalk 节点和应用程序节点。
2. 在 AWS Explorer 中，双击您的 Elastic Beanstalk 环境。
3. 在窗格底部，单击 Events（事件）选项卡。

   此时会显示该应用程序的所有环境的事件列表。

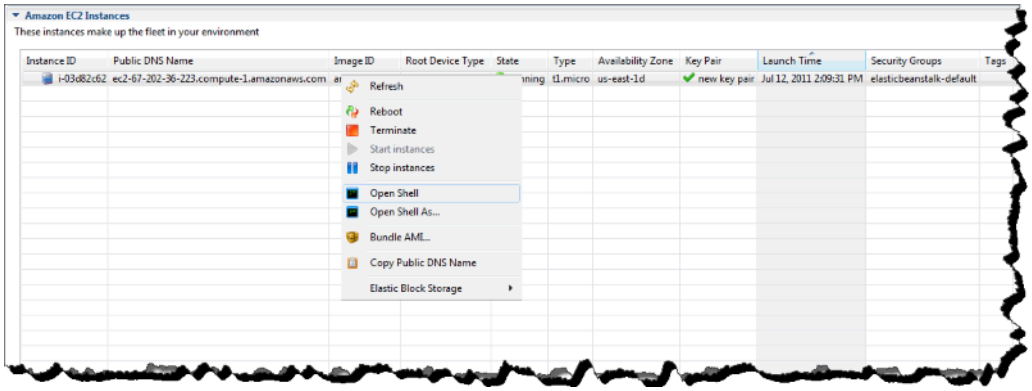


列出和连接到服务器实例

您可以通过 AWS Toolkit for Eclipse 或从 AWS 管理控制台查看运行 Elastic Beanstalk 应用程序环境的 Amazon EC2 实例的列表。您可以使用安全外壳 (SSH) 连接这些实例。有关使用 AWS 管理控制台列出和连接到服务器实例的详细信息，请参阅列出和连接到服务器实例。以下部分逐步介绍了如何使用 AWS Toolkit for Eclipse 查看和连接服务器实例。

查看和连接到环境的 Amazon EC2 实例

1. 在 AWS Toolkit for Eclipse 中，单击 AWS Explorer。展开 Amazon EC2 节点，然后双击 Instances（实例）。
2. 在 Amazon EC2 实例窗口的 Instance ID（实例 ID）列中，右键单击正在您的应用程序负载均衡器中运行的 Amazon EC2 实例的 Instance ID（实例 ID）。然后单击 Open Shell（打开外壳）。



Eclipse 会自动打开 SSH 客户端，并连接到 EC2 实例。

有关连接到 Amazon EC2 实例的更多信息，请参阅 Amazon Elastic Compute Cloud 入门指南。

终止环境

为了避免因为未使用的 AWS 资源而产生费用，您可以使用 AWS Toolkit for Eclipse 终止正在运行的环境。有关环境终止的详细信息，请参阅终止 Elastic Beanstalk 环境。

终止环境

1. 在 AWS Toolkit for Eclipse 中，单击 AWS Explorer 窗格。展开 Elastic Beanstalk 节点。
2. 展开 Elastic Beanstalk 应用程序，然后右键单击 Elastic Beanstalk 环境。
3. 单击 Terminate Environment（终止环境）。Elastic Beanstalk 需要几分钟时间才能终止环境中运行的 AWS 资源。

资源

在开发 Java 应用程序时，您可以在多个地方获取额外的帮助。

资源	描述
AWS Java 开发论坛	发布您的问题并获取反馈。
Java 开发人员中心	示例代码、文档、工具和其他资源的一站式商店。

使用 .NET Core on Linux

> **在 AWS 开发者中心查看.NET**
>
> 你来过我们的 .Net 开发者中心了吗？ 这是我们提供.NET 上所有内容的一站式商店 AWS。有关更多信息，请参阅AWS 开发人员中心上的.NET。

AWS Elastic Beanstalk Linux 上的.NET 内核让您可以使用 Amazon Web Services 轻松部署、管理和扩展您的 Web 应用程序。本章介绍如何将.NET Core Web 应用程序部署到亚马逊 Linux 环境上的 Elastic Beanstalk。您可以使用 Elastic Beanstalk 命令行界面 (EB CLI) 或使用 Elastic Beanstalk 控制台在短短几分钟内部署应用程序。

按照中的步骤使用 EB CLI 创建和部署新的 ASP.NET Core Web 应用程序。QuickStart 适用于 Linux 上的.NET

主题

- QuickStart: 在 Linux 应用程序上部署.NET 核心到 Elastic Beanstalk
- 设置 .NET Core on Linux 开发环境
- 使用 .NET Core on Linux 平台
- AWS Toolkit for Visual Studio – 使用 .Net Core
- 从 .NET on Windows Server 平台迁移到 .NET Core on Linux 平台

QuickStart: 在 Linux 应用程序上部署.NET 核心到 Elastic Beanstalk

本 QuickStart 教程将引导您完成在 Linux 应用程序上创建.NET Core 并将其部署到 AWS Elastic Beanstalk 环境的过程。

> **Note**
>
> 本 QuickStart 教程仅用于演示目的。请勿将本教程中创建的应用程序用于生产流量。

Sections

你的 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

如果您已经有一个 AWS 帐户，则可以继续前进先决条件 。

创建一个 AWS 账户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

先决条件

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

EB CLI

本教程使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

.NET Core on Linux

如果您的本地计算机上没有安装.NET SDK，则可以通过.NET 文档网站上的 “下载.NE T” 链接进行安装。

运行以下命令验证您的.NET SDK 安装情况。

```
~$ dotnet --info
```

步骤 1：在 Linux 应用程序上创建.NET 内核

创建项目目录。

```
~$ mkdir eb-dotnetcore
~$ cd eb-dotnetcore
```

接下来，通过运行以下命令创建一个示例 Hello World 应用程序。

```
~/eb-dotnetcore$ dotnet new web --name HelloElasticBeanstalk
~/eb-dotnetcore$ cd HelloElasticBeanstalk
```

步骤 2：在本地运行应用程序

运行以下命令以在本地运行应用程序。

```
~/eb-dotnetcore/HelloElasticBeasntalk$ dotnet run
```

输出应类似于以下文本。

```
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7294
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5052
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
```

> **Note**
>
> 在本地运行应用程序时，该`dotnet`命令会随机选择一个端口。在本示例中，端口为 5052。当您将应用程序部署到 Elastic Beanstalk 环境时，该应用程序将在端口 5000 上运行。

`http://localhost:`*`port`*在您的网络浏览器中输入 URL 地址。对于这个具体的示例，命令是`http://localhost:5052`。网络浏览器应显示 “Hello World！”。

步骤 3：使用 EB CLI 在 Linux 应用程序上部署.NET 核心

运行以下命令为此应用程序创建 Elastic Beanstalk 环境。

在 Linux 应用程序上创建环境并部署您的.NET Core

1. 编译您的应用程序并将其发布到一个文件夹，以便部署到您即将创建的 Elastic Beanstalk 环境中。

```
~$ cd eb-dotnetcore/HelloElasticBeanstalk
~/eb-dotnetcore/HelloElasticBeanstalk$ dotnet publish -o site
```

2. 导航到您刚刚发布应用程序的网站目录。

```
~/eb-dotnetcore/HelloElasticBeanstalk$ cd site
```

3. 使用 eb init 命令，初始化 EB CLI 存储库。

   请注意有关您在命令中指定的平台分支版本的以下详细信息：

   - 将以下命令替换为 *x.y.z* AL2023 上最新版本的平台分支.NET 6。
   - 要查找最新的平台分支版本，请参阅《平台指南》中的 Linux 支持平台上的.NET Core。AWS Elastic Beanstalk
   - 包含版本号的解决方案堆栈名称的示例是64bit-amazon-linux-2023-v**3.1.1**-running-.net-6。在此示例中，分支版本为 3.1.1。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb init -p 64bit-amazon-linux-2023-
vx.y.z-running-.net-6 dotnetcore-tutorial --region us-east-2
Application dotnetcore-tutorial has been created.
```

   此命令创建名为的应用程序，dotnetcore-tutorial并将您的本地存储库配置为使用命令中指定的.NET Core 在 Linux 平台上创建环境。

4. （可选）再次运行 eb init 以配置默认密钥对，以便使用 SSH 连接到运行您的应用程序的 EC2 实例。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb init
Do you want to set up SSH for your instances?
(y/n): y
Select a keypair.
1) my-keypair
2) [ Create new KeyPair ]
```

如果您已有密钥对，请选择一个，或按提示创建一个。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。

5. 创建环境并使用 eb create 将应用程序部署到此环境中。Elastic Beanstalk 会自动为您的应用程序生成一个 zip 文件并在端口 5000 上启动该文件。

   到

   ```
   ~eb-dotnetcore/HelloElasticBeanstalk/site$ eb create dotnet-tutorial
   ```

   Elastic Beanstalk 创建您的环境大约需要五分钟。

第 4 步：在 Elastic Beanstalk 上运行你的应用程序

创建环境的过程完成后，使用打开您的网站eb open。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb open
```

恭喜您！你已经使用 Elastic Beanstalk 在 Linux 应用程序上部署了.NET 内核！这将使用为应用程序创建的域名打开一个浏览器窗口。

第 5 步：清理

完成应用程序的使用后，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源。

要使用 EB CLI 终止您的 Elastic Beanstalk 环境，请运行以下命令。

```
~eb-dotnetcore/HelloElasticBeanstalk/site$ eb terminate
```

AWS 您的应用程序的资源

您刚刚创建了一个单实例应用程序。它可用作带有单个 EC2 实例的简单示例应用程序，因此不需要负载平衡或 auto Scaling。对于单实例应用程序，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

  各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。

- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 管理所有这些资源。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。您还可以使用 Elastic Beanstalk 控制台探索您的新环境。有关详细步骤，请参阅本指南入门一章中的探索您的环境。

部署一两个示例应用程序并准备好开始在 Linux 应用程序上本地开发和运行.NET Core 之后，请参阅设置 .NET Core on Linux 开发环境。

使用 Elastic Beanstalk 控制台进行部署

您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。

设置 .NET Core on Linux 开发环境

设置 .NET Core 开发环境以在本地测试应用程序，然后再将它部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

有关适用于所有语言的常见设置步骤和工具，请参阅配置用于 Elastic Beanstalk 的开发计算机。

小节目录

- 安装 AWS Toolkit for Visual Studio

安装 .NET Core 开发工具包

您可以使用 .NET Core 开发工具包来开发在 Linux 上运行的应用程序。

请访问 .NET 下载页面，以下载并安装 .NET Core 开发工具包。

安装 IDE

集成开发环境 (IDE) 提供了有助于应用程序开发的大量功能。如果您尚未使用 IDE 进行 .NET 开发，请尝试 Visual Studio Community 来开始操作。

请访问 Visual Studio Community 页面，以下载并安装 Visual Studio Community。

安装 AWS Toolkit for Visual Studio

AWS Toolkit for Visual Studio 是适用于 Visual Studio IDE 的开源插件，能够让开发人员更轻松地使用 AWS开发、调试和部署 .NET 应用程序。有关安装说明，请参阅 Toolkit for Visual Studio 主页。

使用 .NET Core on Linux 平台

AWS Elastic Beanstalk .NET Core on Linux 平台是在 Linux 操作系统上运行的 .NET Core 应用程序的一组平台版本。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅 the section called “扩展 Linux 平台”。以下是一些特定于平台的注意事项。

.NET Core on Linux 平台简介

代理服务器

Elastic Beanstalk .NET Core on Linux 平台包括一个可将请求转发到应用程序的反向代理。默认情况下，Elastic Beanstalk 使用 nginx 作为代理服务器。您可以选择不使用代理服务器，并将 Kestrel 配置为 Web 服务器。默认情况下，Kestrel 包含在 ASP.NET Core 项目模板中。

应用程序结构

您可以发布使用 Elastic Beanstalk 提供的 .NET Core 运行时的运行时依赖应用程序。您还可以发布包含 .NET Core 运行时和源代码包中的应用程序依赖项的自包含应用程序。要了解更多信息，请参阅 the section called “捆绑应用程序”。

平台配置

要配置在您的环境中的服务器实例上运行的进程，请在您的源包中包含一个可选的 Procfile。如果您的源代码包中有多个应用程序，则 `Procfile` 是必需的。

建议您始终在源代码包中将 `Procfile` 与应用程序一起提供。通过这种方式，您可以精确地控制 Elastic Beanstalk 为应用程序运行的进程。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

配置 .NET Core on Linux 环境

您可以通过 .NET Core on Linux 平台设置微调 Amazon EC2 实例的行为。您可以使用 Elastic Beanstalk 控制台编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，并配置应用程序可从环境中读取的变量。

使用 Elastic Beanstalk 控制台配置 .NET Core on Linux 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

日志选项

日志选项部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

环境属性

在 Environment Properties（环境属性）部分中，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。环境属性会以密钥值对的形式传递到应用程序。

在运行于 Elastic Beanstalk 中的 .NET Core on Linux 内，可通过使用 `Environment.GetEnvironmentVariable("variable-name")` 访问环境变量。例如，可以使用以下代码将名为 `API_ENDPOINT` 的属性读取到某个变量。

```
string endpoint = Environment.GetEnvironmentVariable("API_ENDPOINT");
```

参阅 环境属性和其他软件设置 了解更多信息。

.NET Core on Linux 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间中。

除了所有 Elastic Beanstalk 环境支持的选项外，.NET Core on Linux 平台还支持以下命名空间中的选项：

- `aws:elasticbeanstalk:environment:proxy` – 选择使用 nginx 或不使用代理服务器。有效值为 `nginx` 或 `none`。

以下示例配置文件展示了对特定于 .NET Core on Linux 的配置选项的使用：

Example .ebextensions/proxy-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: none
```

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

捆绑适用于 .NET Core on Linux 平台的应用程序

您可以在 AWS Elastic Beanstalk 上运行运行时依赖和自包含 .NET Core 应用程序。

依赖于运行时的应用程序使用 Elastic Beanstalk 提供的 .NET Core 运行时来运行应用程序。Elastic Beanstalk 使用源包中的 `runtimeconfig.json` 文件来确定用于应用程序的运行时。Elastic Beanstalk 选择可用于应用程序的最新兼容运行时。

自包含应用程序包含 .NET Core 运行时、您的应用程序及其依赖项。要使用 Elastic Beanstalk 平台中未包含的 .NET Core 运行时版本，请提供自包含应用程序。

示例

您可以使用 `dotnet publish` 命令编译自包含应用程序和运行时依赖应用程序。有关如何发布 .NET Core 应用程序的更多信息，请参阅 .NET Core 文档中的 .NET Core 应用程序发布概述。

以下示例文件结构定义了使用 Elastic Beanstalk 提供的 .NET Core 运行时的单个应用程序。

```
### appsettings.Development.json
### appsettings.json
### dotnetcoreapp.deps.json
### dotnetcoreapp.dll
### dotnetcoreapp.pdb
### dotnetcoreapp.runtimeconfig.json
### web.config
### Procfile
### .ebextensions
### .platform
```

您可以在源代码包中包含多个应用程序。以下示例定义了在同一 Web 服务器上运行的两个应用程序。要运行多个应用程序，您必须在源代码包中包含 Procfile。如需查看完整的示例应用程序，请参见 dotnet-core-linux-multiple-apps.zip。

```
### DotnetMultipleApp1
#   ### Amazon.Extensions.Configuration.SystemsManager.dll
#   ### appsettings.Development.json
#   ### appsettings.json
#   ### AWSSDK.Core.dll
#   ### AWSSDK.Extensions.NETCore.Setup.dll
#   ### AWSSDK.SimpleSystemsManagement.dll
#   ### DotnetMultipleApp1.deps.json
```

```
#   ### DotnetMultipleApp1.dll
#   ### DotnetMultipleApp1.pdb
#   ### DotnetMultipleApp1.runtimeconfig.json
#   ### Microsoft.Extensions.PlatformAbstractions.dll
#   ### Newtonsoft.Json.dll
#   ### web.config
### DotnetMultipleApp2
#   ### Amazon.Extensions.Configuration.SystemsManager.dll
#   ### appsettings.Development.json
#   ### appsettings.json
#   ### AWSSDK.Core.dll
#   ### AWSSDK.Extensions.NETCore.Setup.dll
#   ### AWSSDK.SimpleSystemsManagement.dll
#   ### DotnetMultipleApp2.deps.json
#   ### DotnetMultipleApp2.dll
#   ### DotnetMultipleApp2.pdb
#   ### DotnetMultipleApp2.runtimeconfig.json
#   ### Microsoft.Extensions.PlatformAbstractions.dll
#   ### Newtonsoft.Json.dll
#   ### web.config
### Procfile
### .ebextensions
### .platform
```

使用 Procfile 配置 .NET Core on Linux 环境

要在同一 Web 服务器上运行多个应用程序，您必须在源代码包中包含 `Procfile`，以告知 Elastic Beanstalk 要运行哪些应用程序。

建议您始终在源代码包中将 `Procfile` 与应用程序一起提供。通过这种方式，您可以精确控制 Elastic Beanstalk 为您的应用程序运行的进程以及这些进程接收的参数。

以下示例使用 `Procfile` 为 Elastic Beanstalk 指定要在同一 Web 服务器上运行的两个应用程序。

Example Procfile

```
web: dotnet ./dotnet-core-app1/dotnetcoreapp1.dll
web2: dotnet ./dotnet-core-app2/dotnetcoreapp2.dll
```

有关编写和使用 `Procfile` 的详细信息，请展开the section called “扩展 Linux 平台”中的 Buildfile 和 Procfile 部分 。

为 .NET Core on Linux 环境配置代理服务器

AWS Elastic Beanstalk 使用 nginx 作为反向代理，将请求中继到您的应用程序。Elastic Beanstalk 提供一个默认 nginx 配置，您可以扩展该配置，也可以使用您自己的配置完全覆盖该配置。

默认情况下，Elastic Beanstalk 将 nginx 代理配置为通过端口 5000 向您的应用程序转发请求。您可以覆盖默认端口，方法是将 PORT 环境属性设置为主应用程序侦听的端口。

> **Note**
>
> 应用程序侦听的端口不会影响 nginx 服务器为了从负载均衡器接收请求而侦听的端口。

在平台版本上配置代理服务器

所有 AL2023/AL2 平台都支持统一的代理配置功能。有关在运行 AL2023/AL2 的平台版本上配置代理服务器的更多信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

以下示例配置文件扩展了环境的 nginx 配置。该配置会将发往 `/api` 的请求定向到侦听 Web 服务器上的 5200 端口的第二个 Web 应用程序。默认情况下，Elastic Beanstalk 会将请求转发到侦听 5000 端口的单个应用程序。

Example **`01_custom.conf`**

```
location /api {
     proxy_pass          http://127.0.0.1:5200;
     proxy_http_version  1.1;

     proxy_set_header   Upgrade $http_upgrade;
     proxy_set_header   Connection $http_connection;
     proxy_set_header   Host $host;
     proxy_cache_bypass $http_upgrade;
     proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
     proxy_set_header   X-Forwarded-Proto $scheme;
}
```

AWS Toolkit for Visual Studio – 使用 .Net Core

AWS Toolkit for Visual Studio 是 Visual Studio IDE 的插件。使用该工具包，您在 Visual Studio 环境中工作时，可以在 Elastic Beanstalk 中部署和管理应用程序。

本主题说明如何使用 AWS Toolkit for Visual Studio 执行以下任务：

- 使用 Visual Studio 模板创建 ASP.NET Core Web 应用程序。
- 创建 Elastic Beanstalk Amazon Linux 环境。
- 将 ASP.NET Core Web 应用程序部署到新的 Amazon Linux 环境。

本主题还探讨如何使用 AWS Toolkit for Visual Studio 管理 Elastic Beanstalk 应用程序环境和监控应用程序的运行状况。

小节目录

先决条件

在开始本教程之前，您需要安装 AWS Toolkit for Visual Studio。有关说明，请参阅设置 AWS Toolkit for Visual Studio。

如果您之前从未用过此工具包，则在安装此工具包后首先需要使用此工具包注册您的AWS凭证。有关这一点的更多信息，请参阅提供AWS凭证。

创建新的应用程序项目

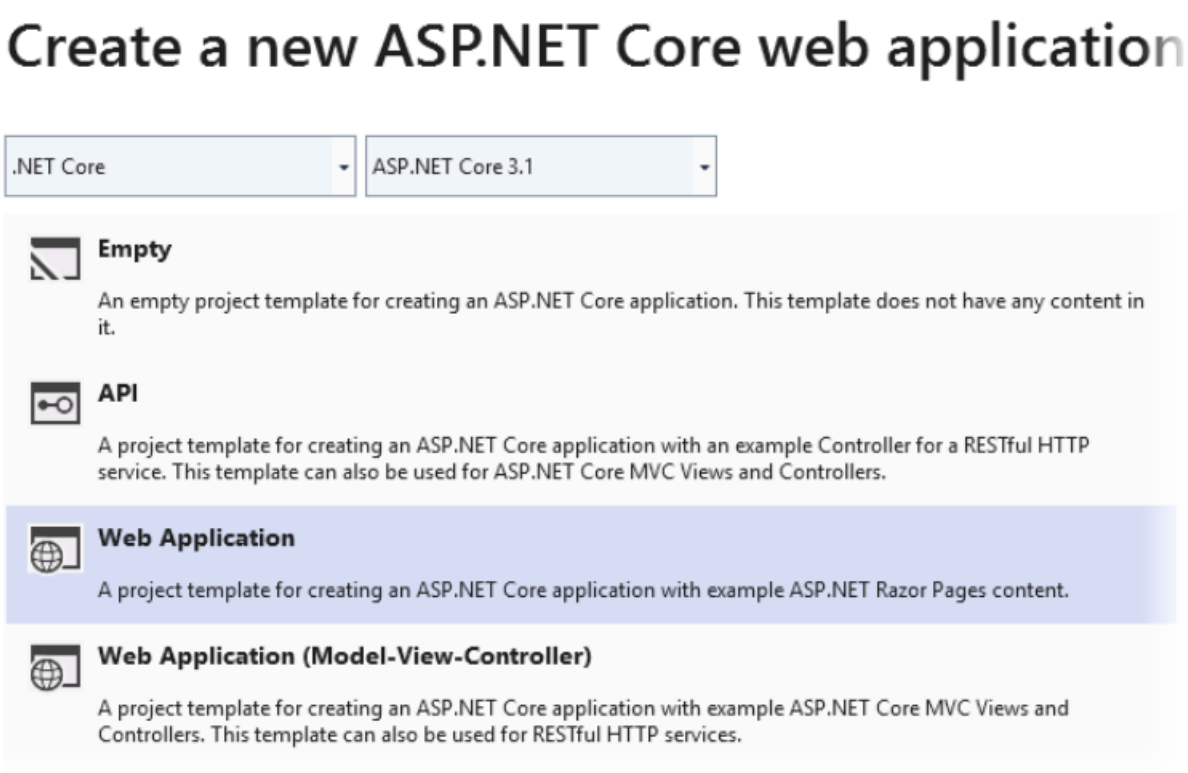
如果您在 Visual Studio 中没有 .NET Core 应用程序项目，则可以使用其中一个 Visual Studio 项目模板轻松创建一个项目。

创建新的 ASP.NET Core Web 应用程序项目

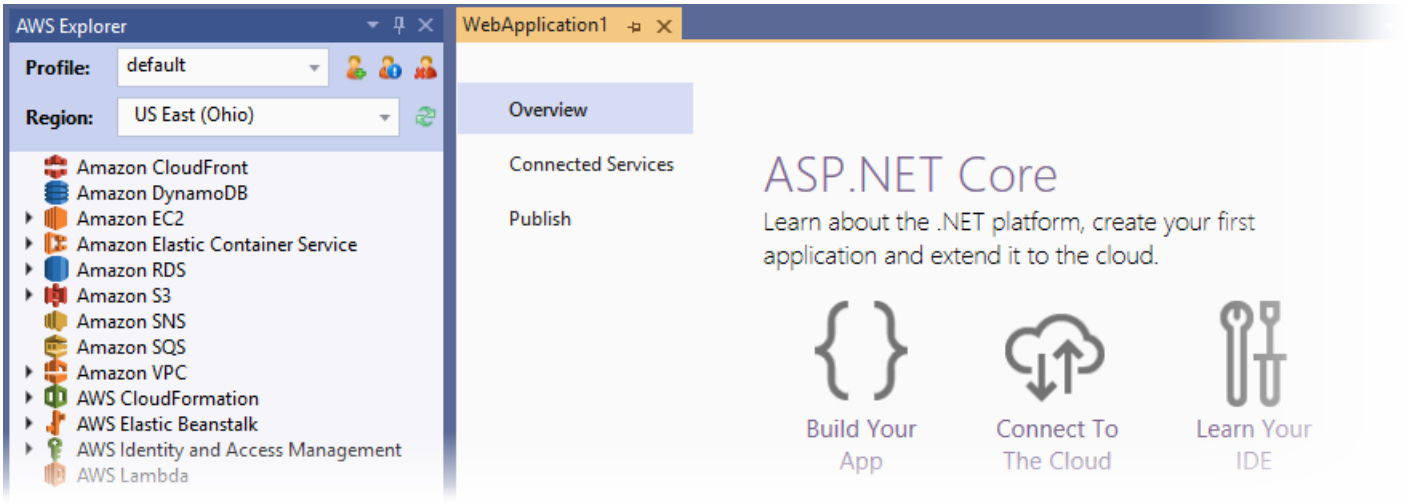
1. 在 Visual Studio 中，在 File（文件）菜单上选择 New（新建），然后选择 Project（项目）。
2. 在创建新项目对话框中，选择 C#，选择 Linux，然后选择云。
3. 从显示的项目模板列表中选择 ASP.NET Core Web 应用程序，然后选择下一步。

> **Note**
>
> 如果您在项目模板中没有看到 ASP.NET Core Web Application（ASP.NET Core Web 应用程序）列出，则可以在 Visual Studio 中安装它。
>
> 1. 滚动到模板列表的底部，然后选择位于模板列表下的安装更多工具和功能链接。
> 2. 如果系统提示您允许 Visual Studio 应用程序对设备进行更改，请选择是。
> 3. 选择工作负载选项卡，然后选择 ASP.NET 和 Web 开发。
> 4. 选择修改按钮。Visual Studio 安装程序将安装项目模板。
> 5. 安装程序完成后，退出面板以返回到 Visual Studio 中您之前离开的位置。

4. 在配置新项目对话框中，输入项目名称。解决方案名称默认为您的项目名称。接下来，选择创建。
5. 在创建新的 ASP.NET Core Web 应用程序对话框中，选择 .NET Core，然后选择 ASP.NET Core 3.1。从显示的应用程序类型列表中选择 Web 应用程序，然后选择创建按钮。



Visual Studio 在创建应用程序时将显示 Creating Project（正在创建项目）对话框。当 Visual Studio 完成生成应用程序后，将显示一个包含您的应用程序名称的面板。

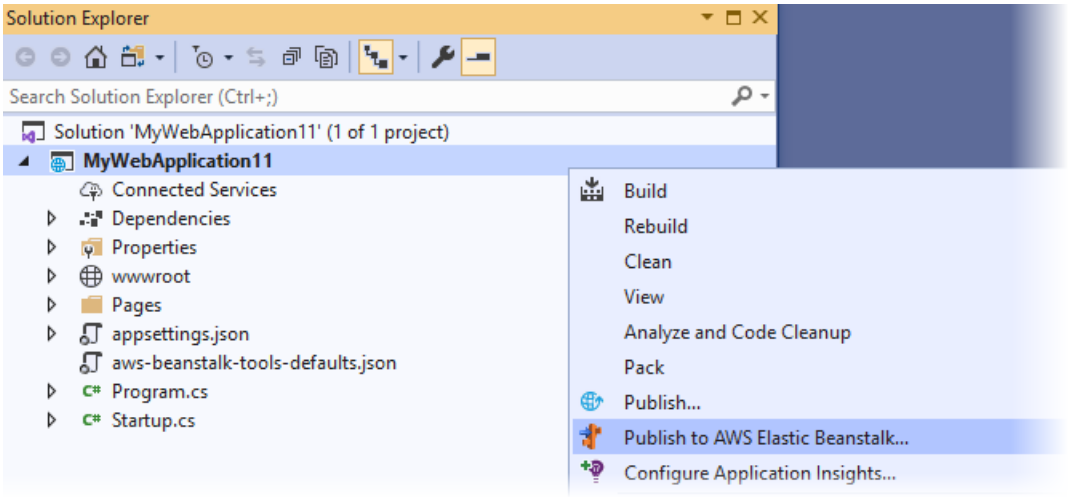


创建 Elastic Beanstalk 环境并部署应用程序

本节介绍如何为应用程序创建 Elastic Beanstalk 环境并将应用程序部署到该环境。

创建新环境并部署应用程序

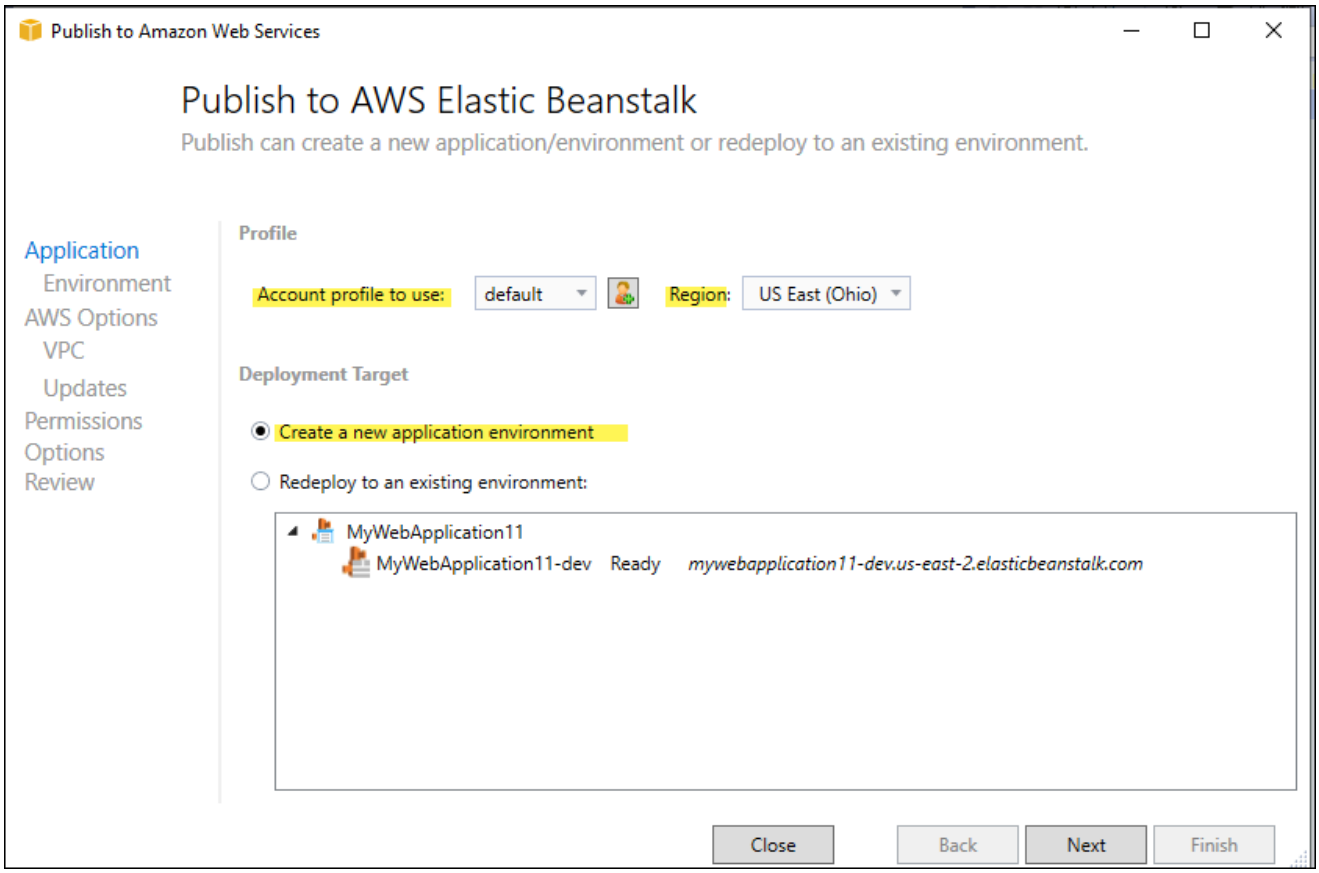
1. 在 Visual Studio 中，依次选择 View（查看）和 Solution Explorer（解决方案浏览器）。
2. 在 Solution Explorer（解决方案资源管理器）中，打开应用程序的上下文（右键单击）菜单，然后选择 Publish to AWS Elastic Beanstalk（发布到 Amazon Elastic Beanstalk）。



3. 在 Publish to AWS Elastic Beanstalk（发布到亚马逊云科技）向导中，输入您的账户信息。
   a. 对于要使用的账户配置文件，请选择默认账户或选择添加其他账户图标以输入新账户信息。

b. 对于区域，选择要在其中部署应用程序的区域。有关可用 AWS 区域的信息，请参阅 AWS 一般参考 的 AWS Elastic Beanstalk 端点和配额。如果您选择了一个 Elastic Beanstalk 不支持的区域，则部署到 Elastic Beanstalk 的选项不可用。

c. 选择创建新的应用程序环境，然后选择下一步。



4. 在应用程序环境对话框中，输入新应用程序环境的详细信息。

5. 在下一个AWS选项对话框中，为已部署的应用程序设置 Amazon EC2 选项和其他AWS相关选项。

a. 对于容器类型，选择运行 .NET Core 的 64 位 Amazon Linux 2 v*<n.n.n>*。

> **Note**
>
> 我们建议您选择 Linux 的当前平台版本。此版本包含最新 Amazon Machine Image (AMI) 中包含的最新安全和错误修复。

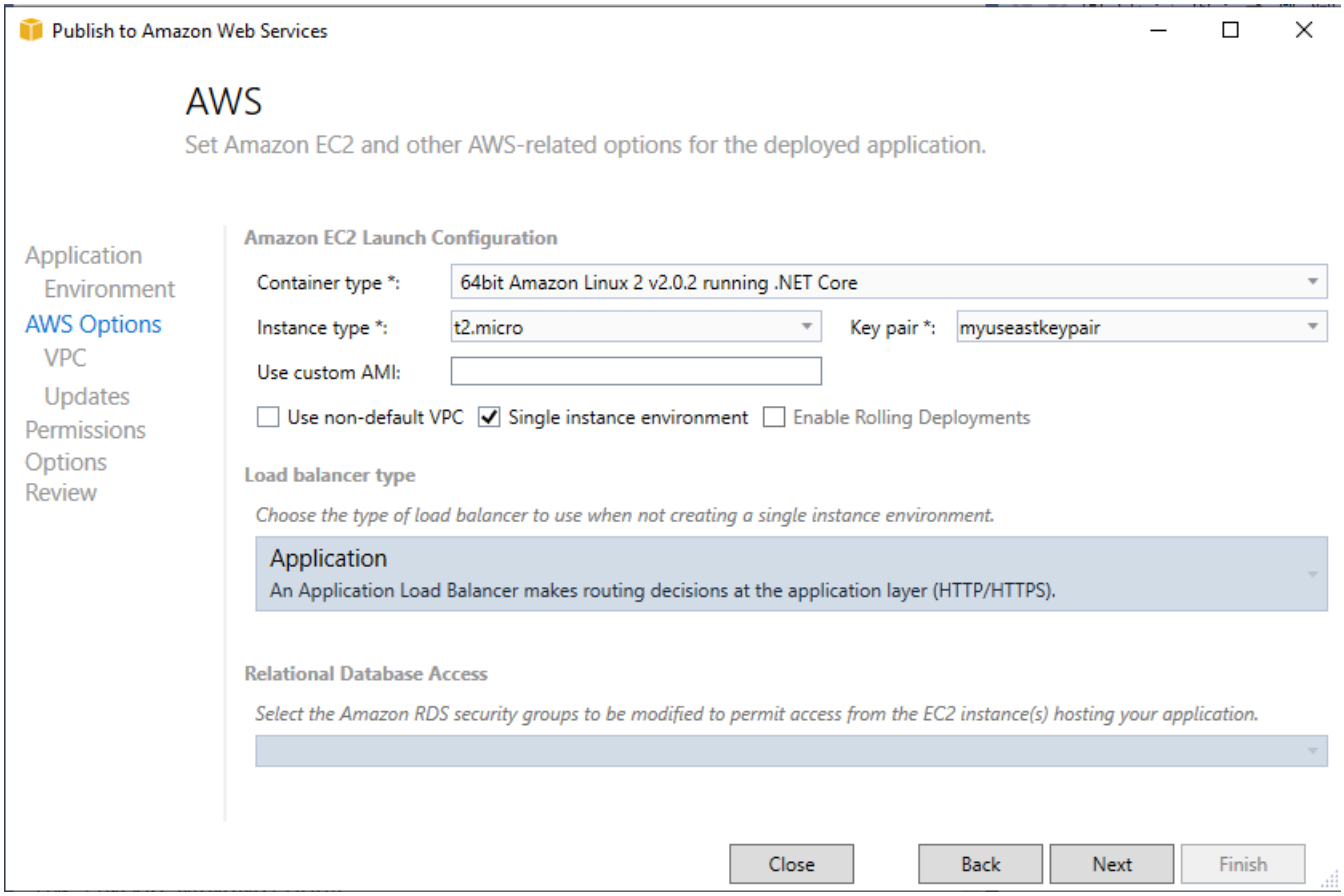
b. 对于实例类型，选择 t2.micro。（选择微型实例类型将最大限度地降低与运行实例相关的成本。）

c. 对于 Key pair（密钥对），选择 Create new key pair（创建新密钥对）。输入新密钥对的名称，然后选择确定。（在此示例中，我们使用 **myuseastkeypair**）。利用密钥对，可

以对 Amazon EC2 实例进行远程桌面访问。有关 Amazon EC2 密钥对的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南中的使用凭证。

d. 对于简单的低流量应用程序，请选择单一实例环境。有关更多信息，请参阅 环境类型

e. 选择 Next（下一步）。



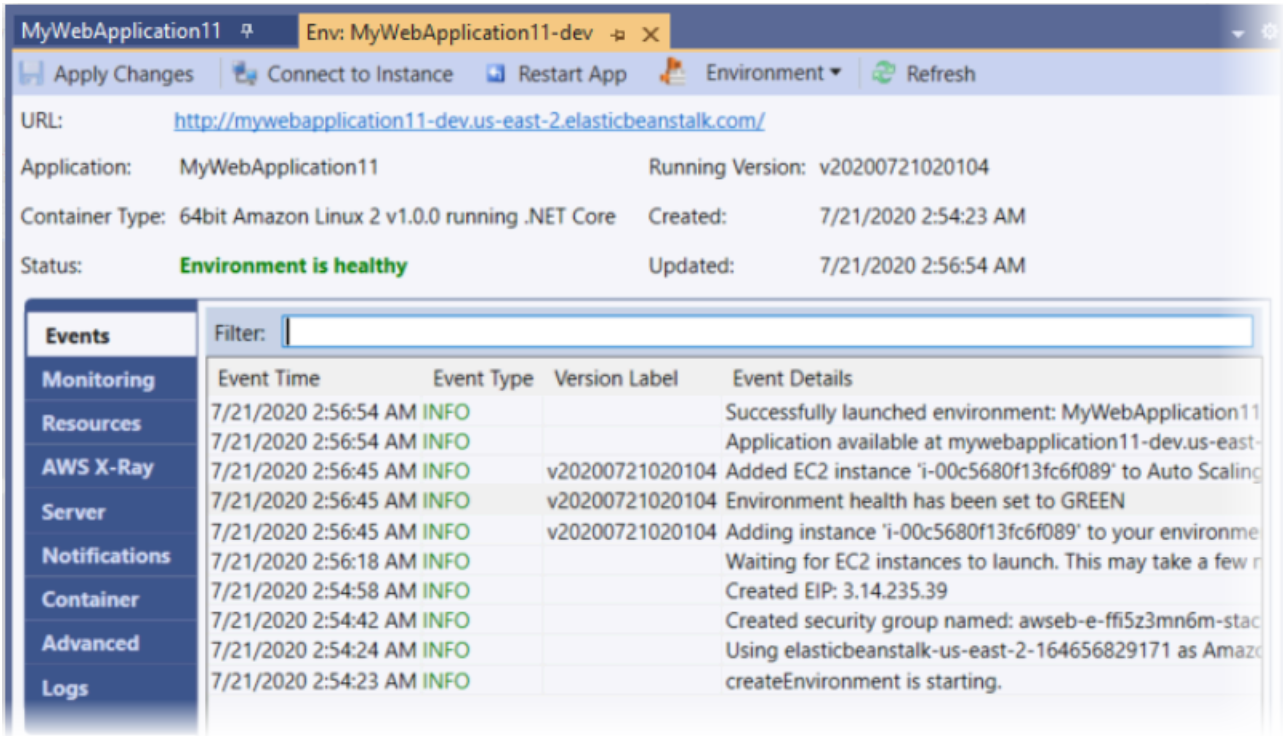
有关本示例中未使用的AWS选项的更多信息，请考虑以下页面：

- 有关使用自定义 AMI，请参阅使用自定义 Amazon 系统映像 (AMI)。
- 如果您未选择单一实例环境，则需要选择负载均衡类型。参阅 Elastic Beanstalk 环境的负载均衡器 了解更多信息。
- 如果您没有选择 Use non-default VPC（使用非默认 VPC），Elastic Beanstalk 将使用默认 Amazon VPC（Amazon Virtual Private Cloud）配置。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon VPC 结合使用。
- 通过选择启用滚动部署选项，可以将部署拆分为多个批处理，以避免部署期间可能停机。有关更多信息，请参阅将应用程序部署到 Elastic Beanstalk 环境。

- 通过选择 Relational Database Access（关系数据库访问）选项，可以将 Elastic Beanstalk 环境连接到以前创建的具有 Amazon RDS 数据库安全组的 Amazon RDS 数据库。有关更多信息，请参阅 Amazon RDS 用户指南中的使用安全组控制访问权限。

6. 在权限对话框中选择下一步。
7. 在应用程序选项对话框中选择下一步。
8. 查看您的部署选项。验证设置是否正确后，选择部署。

您的 ASP.NET Core Web 应用程序将导出为 Web 部署文件。此文件会上传到 Amazon S3，并通过 Elastic Beanstalk 注册为一个新的应用程序版本。Elastic Beanstalk 部署功能会监控您的现有环境，直到该环境可用且具有最新部署的代码。“环境:<环境名称>”选项卡上将显示环境的状态。状态更新为环境运行状况正常后，您可以选择要启动 Web 应用程序的 URL 地址。



终止环境

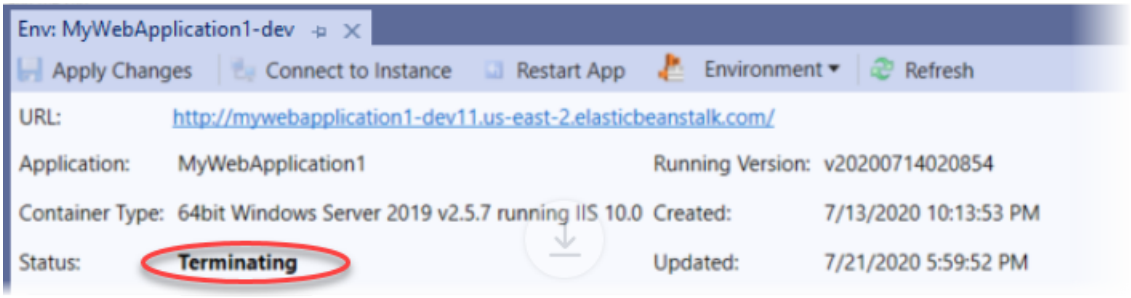
为避免未使用的AWS资源产生费用，可以使用 AWS Toolkit for Visual Studio 终止正在运行的环境。

> **Note**
>
> 稍后，您始终都可以使用相同的版本启动新的环境。

终止环境

1. 展开 Elastic Beanstalk 节点和应用程序节点。在 AWS Explorer 中，打开应用程序环境的上下文（右键单击）菜单，然后选择 Terminate Environment（终止环境）。
2. 当系统提示时，选择是以确认要终止该环境。Elastic Beanstalk 需要几分钟时间才能终止环境中运行的 AWS 资源。

“环境:<环境变量>”选项卡上环境的状态将更改为正在终止，并最终更改为已终止。



> **Note**
>
> 终止环境时，与已终止环境相关联的别名记录可供任何人使用。

管理 Elastic Beanstalk 应用程序环境

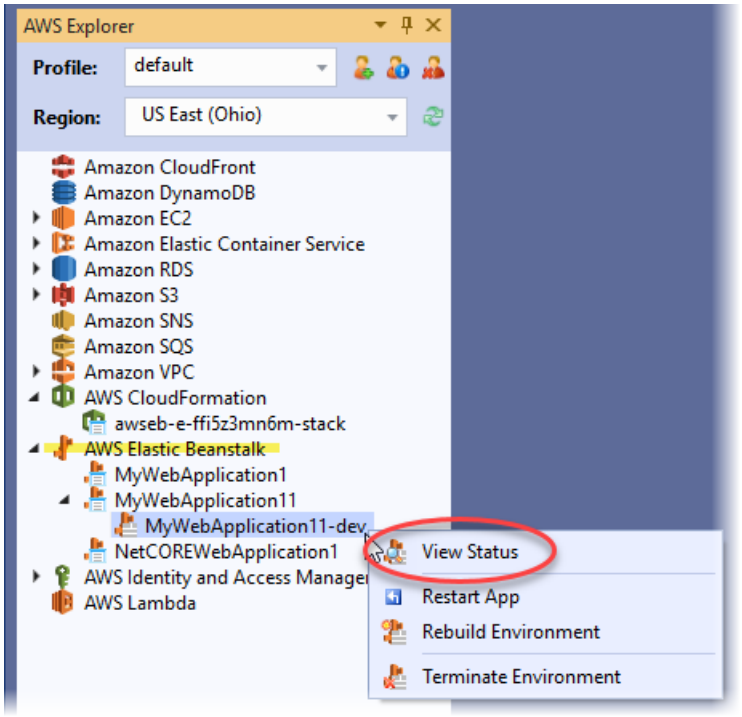
通过 AWS Toolkit for Visual Studio 和 AWS 管理控制台，您可以更改应用程序环境使用的 AWS 资源的预置和配置。有关如何使用 AWS 管理控制台管理应用程序环境的详细信息，请参阅管理环境。本部分介绍可在 AWS Toolkit for Visual Studio 中编辑的特定服务设置，它也是应用程序环境配置的一部分。

更改环境配置设置

在部署您的应用程序时，Elastic Beanstalk 会配置多个已连接的AWS云计算服务。您可以使用 AWS Toolkit for Visual Studio 控制如何配置各项服务。

编辑应用程序的环境设置

1. 在 Visual Studio 中的 File（文件）菜单上，选择 AWS Explorer。
2. 展开 Elastic Beanstalk 节点和应用程序节点。打开应用程序环境的上下文（右键单击）菜单，然后选择查看状态。

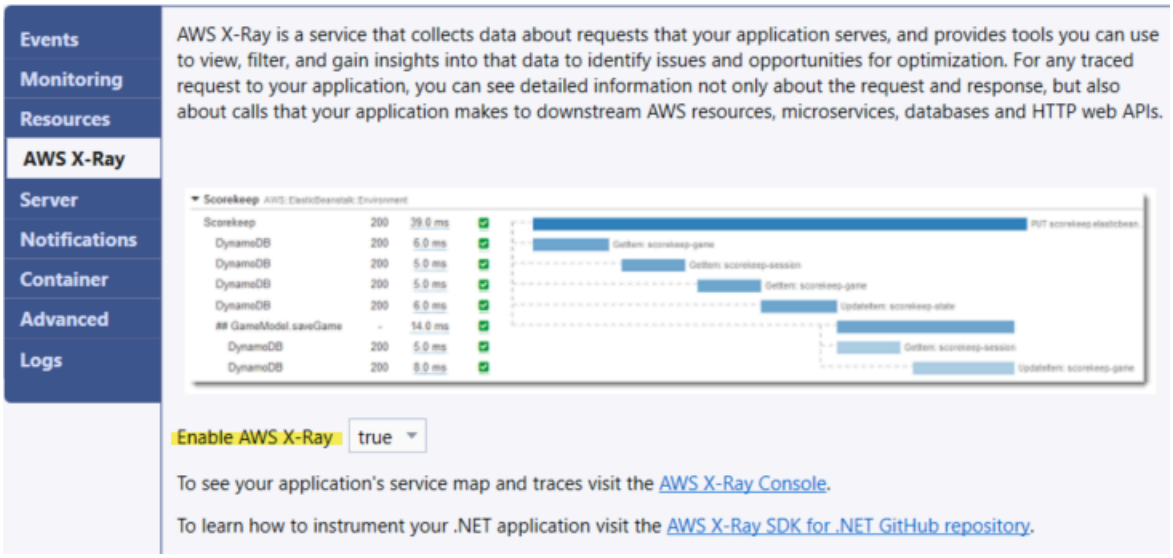


您现在可以配置以下各项的设置：

- AWS X-Ray
- 服务器
- 负载均衡器（仅适用于多实例环境）
- Auto Scaling（仅适用于多实例环境）
- 通知
- 容器
- 高级配置选项

使用 AWS Toolkit for Visual Studio 配置 AWS X-Ray

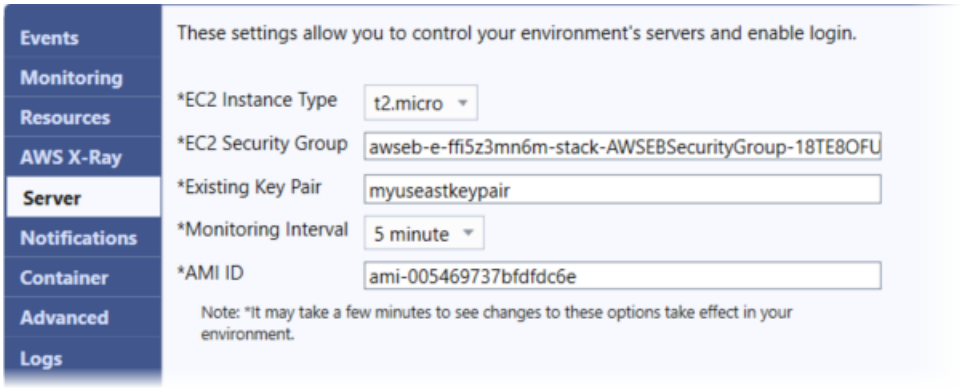
AWS X-Ray 提供请求跟踪、异常收集和分析功能。使用 AWS X-Ray 面板，您可以为应用程序启用或禁用 X-Ray。有关 X-Ray 的更多信息，请参阅 AWS X-Ray 开发人员指南。



使用 AWS Toolkit for Visual Studio 配置 EC2 实例

您可以使用 Amazon Elastic Compute Cloud (Amazon EC2) 启动和管理 Amazon 数据中心内的服务器实例。您可以随时使用 Amazon EC2 服务器实例，并可根据需要在任意长的时间内使用，也可以用于任何合法用途。实例可以按照不同的规模和配置进行提供。有关更多信息，请参阅 Amazon EC2。

您可以使用 AWS Toolkit for Visual Studio 的应用程序环境选项卡内的 Server（服务器）选项卡来编辑 Amazon EC2 实例配置。



Amazon EC2 实例类型

Instance type（实例类型）显示可用于您的 Elastic Beanstalk 应用程序的实例类型。请更改实例类型以选择特征（包括内存大小和 CPU 处理能力）最适合您的应用程序的服务器。例如，具有大量操作和长时间运行的操作的应用程序可能需要更多 CPU 或内存。

有关可用于 Elastic Beanstalk 应用程序的 Amazon EC2 实例类型的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的实例类型。

Amazon EC2 安全组

您可以使用 Amazon EC2 安全组 控制对 Elastic Beanstalk 应用程序的访问。安全组会定义实例的防火墙规则。这些规则会指定应将哪些传入的网络流量提交给实例。所有其他传入流量将被丢弃。您可以随时针对不同的组修改这些规则。新的规则会自动在所有现在运行的和将来启动的实例上强制实施。

您可以指定哪些 Amazon EC2 安全组控制对 Elastic Beanstalk 应用程序的访问。为此，请在 EC2 安全组文本框中输入特定 Amazon EC2 安全组的名称（用逗号分隔多个安全组）。您可以使用AWS管理控制台或 AWS Toolkit for Visual Studio 完成此操作。

使用 AWS Toolkit for Visual Studio 创建安全组

1. 在 Visual Studio 中的 AWS Explorer 中，展开 Amazon EC2 节点，然后选择 Security Groups（安全组）。
2. 选择创建安全组，然后输入安全组的名称和说明。
3. 选择确定。

有关 Amazon EC2 安全组的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的使用安全组。

Amazon EC2 密钥对

您可以使用 Amazon EC2 密钥对安全地登录到为 Elastic Beanstalk 应用程序预配置的 Amazon EC2 实例。

> **Important**
>
> 您必须创建 Amazon EC2 密钥对并配置由 Elastic Beanstalk 预置的 Amazon EC2 实例，以便能够访问这些实例。在向 Elastic Beanstalk 部署您的应用程序时，您可以使用 AWS Toolkit for Visual Studio 内的 Publish to AWS（发布到亚马逊云科技）向导创建密钥对。如果要使用该 Toolkit 创建额外的密钥对，请按照此处所述的步骤操作。也可以使用 AWS 管理控制台设置 Amazon EC2 密钥对。有关为 Amazon EC2 创建密钥对的说明，请参阅 Amazon Elastic Compute Cloud 入门指南。

Existing Key Pair（现有密钥对）文本框可让您指定 Amazon EC2 密钥对的名称，您可以使用该密钥对安全地登录到运行 Elastic Beanstalk 应用程序的 Amazon EC2 实例。

指定 Amazon EC2 密钥对的名称

1. 展开 Amazon EC2 节点并选择密钥对。
2. 选择创建密钥对并输入密钥对名称。
3. 选择确定。

有关 Amazon EC2 密钥对的更多信息，请转到 Amazon Elastic Compute Cloud 用户指南 中的使用 Amazon EC2 凭证。有关连接到 Amazon EC2 实例的更多信息，请参阅

监控间隔

默认情况下，仅启用基本 Amazon CloudWatch 指标。这些指标会以五分钟为周期返回数据。在 AWS Toolkit for Eclipse 中，您可为环境启用更精细的一分钟 CloudWatch 指标，方法为在环境的 Configuration（配置）选项卡的 Server（服务器）部分中为 Monitoring Interval（监控间隔）选择 1 minute（1 分钟）。

> **Note**
>
> 一分钟时间间隔指标可能产生 Amazon CloudWatch 服务费用。有关更多信息，请参阅 Amazon CloudWatch。

自定义 AMI ID

在 AWS Toolkit for Eclipse 中，您可将自定义 AMI 的标识符输入到环境的 Configuration（配置）选项卡的 Server（服务器）部分中的 Custom AMI ID（自定义 AMI ID）框，以便使用您自己的自定义 AMI 覆盖用于 Amazon EC2 实例的默认 AMI。

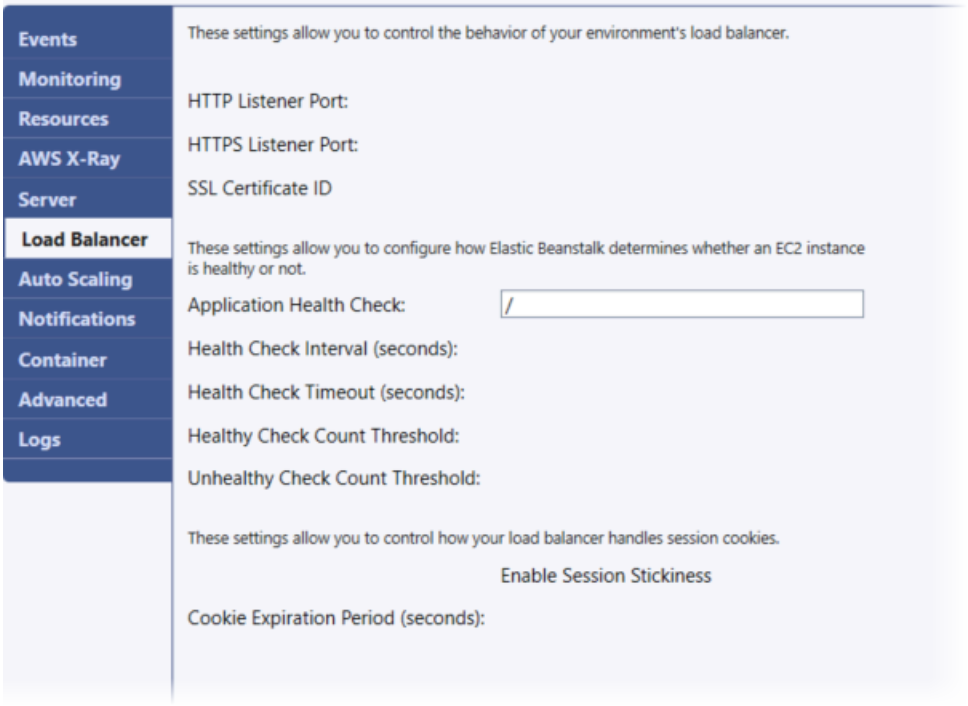
> **Important**
>
> 使用您自己的 AMI 是一项高级任务，应小心谨慎地执行。如果需要自定义 AMI，则建议您先使用默认 Elastic Beanstalk AMI，然后修改它。若要保持正常运行状态，Elastic Beanstalk 期望 Amazon EC2 实例满足一系列要求，包括具有一个正在运行的主机管理器。如果未满足这些要求，您的环境可能无法正常运行。

使用 AWS Toolkit for Visual Studio 配置 Elastic Load Balancing

Elastic Load Balancing 是一种 Amazon Web 服务，可帮助您提高应用程序的可用性和可扩展性。该服务可让您轻松地在两个或更多的 Amazon EC2 实例之间分配应用程序负载。Elastic Load Balancing 通过提供附加冗余来提高可用性，并支持应用程序的流量增长。

使用 Elastic Load Balancing，您可以在所有正在运行的实例间自动分配和平衡传入的应用程序流量。还可以在需要增加应用程序容量时轻松添加新实例。

Elastic Beanstalk 会在您部署应用程序时自动地预配置 Elastic Load Balancing。您可以使用 AWS Toolkit for Visual Studio 的应用程序环境选项卡内的 Load Balancer（负载均衡器）选项卡来编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。



以下部分介绍了可为应用程序配置的 Elastic Load Balancing 参数。

端口

预配置来处理您的 Elastic Beanstalk 应用程序请求的负载均衡器会将请求发送到正在运行您的应用程序的 Amazon EC2 实例。预配置的负载均衡器会侦听 HTTP 和 HTTPS 端口上的请求，并将请求路由到 AWS Elastic Beanstalk 应用程序中的 Amazon EC2 实例。默认情况下，负载均衡器将处理 HTTP 端口上的请求。为此，必须至少打开其中一个端口，要么是 HTTP 要么是 HTTPS。

> **Important**
>
> 确保您指定的端口没有锁定；否则，用户将无法连接到 Elastic Beanstalk 应用程序。

控制 HTTP 端口

若要关闭 HTTP 端口，请为 HTTP Listener Port（HTTP 侦听器端口）选择 OFF（关）。若要打开 HTTP 端口，需从列表中选择一个 HTTP 端口（例如，80）。

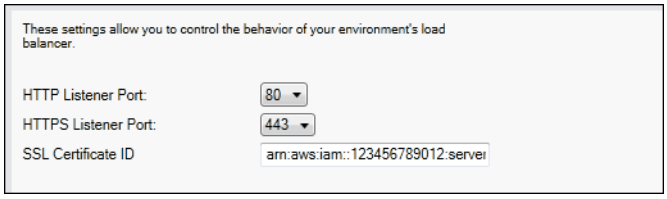
> **Note**
>
> 要使用默认端口 80 (如端口 8080) 以外的端口来访问您的环境，请将侦听器添加到现有负载均衡器并配置新侦听器来侦听该端口。
> 例如，当使用适用于 Classic Load Balancer 的 AWS CLI 时，键入以下命令可将 *LOAD_BALANCER_NAME* 替换为您用于 Elastic Beanstalk 的负载均衡器的名称。
>
> ```
> aws elb create-load-balancer-listeners --load-balancer-name LOAD_BALANCER_NAME
>  --listeners "Protocol=HTTP, LoadBalancerPort=8080, InstanceProtocol=HTTP,
>  InstancePort=80"
> ```
>
> 例如，当使用适用于 Application Load Balancer 的 AWS CLI 时，键入以下命令可将 *LOAD_BALANCER_ARN* 替换为您用于 Elastic Beanstalk 的负载均衡器的 ARN。
>
> ```
> aws elbv2 create-listener --load-balancer-arn LOAD_BALANCER_ARN --protocol HTTP
>  --port 8080
> ```
>
> 如果希望 Elastic Beanstalk 监控您的环境，请勿删除端口 80 上的侦听器。

控制 HTTPS 端口

Elastic Load Balancing 支持 HTTPS/TLS 协议，可为负载均衡器的客户端连接流量加密。从负载均衡器到 EC2 实例的连接使用明文加密。默认情况下，HTTPS 端口是关闭的。

打开 HTTPS 端口

1. 使用 AWS Certificate Manager (ACM) 创建新的证书，或者将证书和密钥上传到 AWS Identity and Access Management (IAM)。有关请求 ACM 证书的更多信息，请参阅 AWS Certificate Manager 用户指南中的请求证书。有关将第三方证书导入 ACM 中的更多信息，请参阅 AWS Certificate Manager 用户指南中的导入证书。如果 ACM 在您所在的区域不可用，请使用 AWS Identity and Access Management (IAM) 上传第三方证书。ACM 和 IAM 服务存储证书并为 SSL 证书提供 Amazon Resource Name (ARN)。有关创建证书并将证书上传到 IAM 的更多信息，请参阅 IAM 用户指南 中的使用服务器证书。
2. 通过为 HTTPS Listener Port（HTTP 侦听器端口）选择端口来指定 HTTPS 端口。



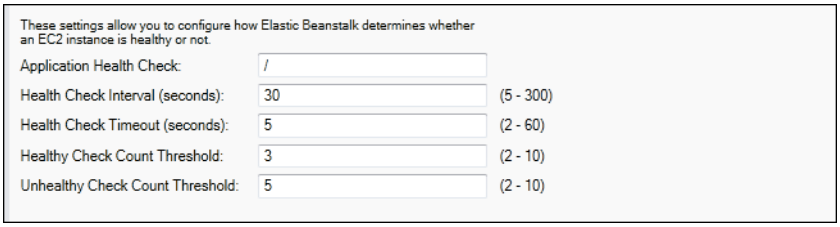
3. 对于 SSL Certificate ID（SSL 证书 ID），输入 SSL 证书的 Amazon Resource Name（ARN）。例如，**`arn:aws:iam::123456789012:server-certificate/abc/certs/build`** 或 **`arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`**。使用您在步骤 1 中创建或上传的 SSL 证书。

若要关闭 HTTPS 端口，请为 HTTPS Listener Port（HTTPS 侦听器端口）选择 OFF（关）。

运行状况检查

运行状况检查定义包括一个要用来查询实例运行状况的 URL。默认情况下，对于非早期容器，Elastic Beanstalk 使用 TCP:80，而对于早期容器，则使用 HTTP:80。您可以通过在应用程序运行状况检查 URL 框中输入 URL（例如 `/myapp/default.aspx`）以覆盖默认 URL，使之与应用程序中的现有资源匹配。如果您覆盖默认 URL，则 Elastic Beanstalk 将使用 HTTP 来查询资源。要检查您使用的是否是早期容器类型，请参阅the section called “为什么某些平台版本标记为传统版本？”。

您可以使用 Load Balancing（负载均衡）面板的 EC2 Instance Health Check（EC2 实例运行状况检查）部分来控制运行状况检查的设置。



运行状况检查定义包括一个要用来查询实例运行状况的 URL。通过在应用程序运行状况检查 URL 框中输入 URL（例如 `/myapp/index.jsp`）以覆盖默认 URL，使之与应用程序中的现有资源匹配。

下表介绍了可为您的应用程序设置的运行状况检查参数。

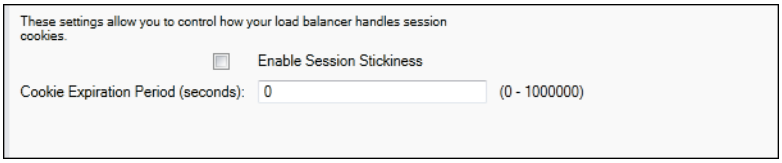
- 对于 Health Check Interval (seconds)（运行状况检查间隔(秒)），输入 Elastic Load Balancing 在对应用程序的 Amazon EC2 实例的运行状况进行检查之间等待的秒数。
- 对于 Health Check Timeout (seconds)（运行状况检查超时(秒)），指定 Elastic Load Balancing 在将实例视为无响应之前等待响应的秒数。
- 对于 Healthy Check Count Threshold（良好运行状况检查计数阈值）和 Unhealthy Check Count Threshold（不佳运行状况检查计数阈值），指定 Elastic Load Balancing 更改实例的运行状况状态之前连续的成功或失败 URL 探测的次数。例如，为运行状况不正常检查计数阈值指定 **5**，即表示必须在该 URL 连续 5 次返回错误消息或超时后，Elastic Load Balancing 才将运行状况检查视为失败。

会话

默认情况下，负载均衡器会以最小的负载将每个请求独立地传送给该服务器实例。比较起来，粘性 (VPC) 会将用户的会话绑定到具体的服务器实例，以便该用户在会话期间发出的所有请求都会发送到同一个服务器实例中。

在为应用程序启用粘性会话后，Elastic Beanstalk 会使用负载均衡器生成的 HTTP Cookie。负载均衡器会使用负载均衡器生成的特别 Cookie 来跟踪每个请求的应用程序实例。在负载均衡器收到请求时，它首先会检查并查看请求中是否存在这个 Cookie。如果存在，该请求会发送到在 Cookie 中指定的应用程序实例。如果没有 Cookie，负载均衡器会根据现有的负载均衡算法选择一个应用程序实例。响应中会插入 Cookie，从而将同一用户发出的后续请求绑定到该应用程序实例中。策略配置会定义 Cookie 的到期时间，从而确定每个 Cookie 的有效持续时间。

您可以使用负载均衡器选项卡上的会话部分指定是否让应用程序的负载均衡器支持会话粘性。



有关 Elastic Load Balancing 的更多信息，请转到 Elastic Load Balancing 开发人员指南。

使用 AWS Toolkit for Visual Studio 配置 Auto Scaling

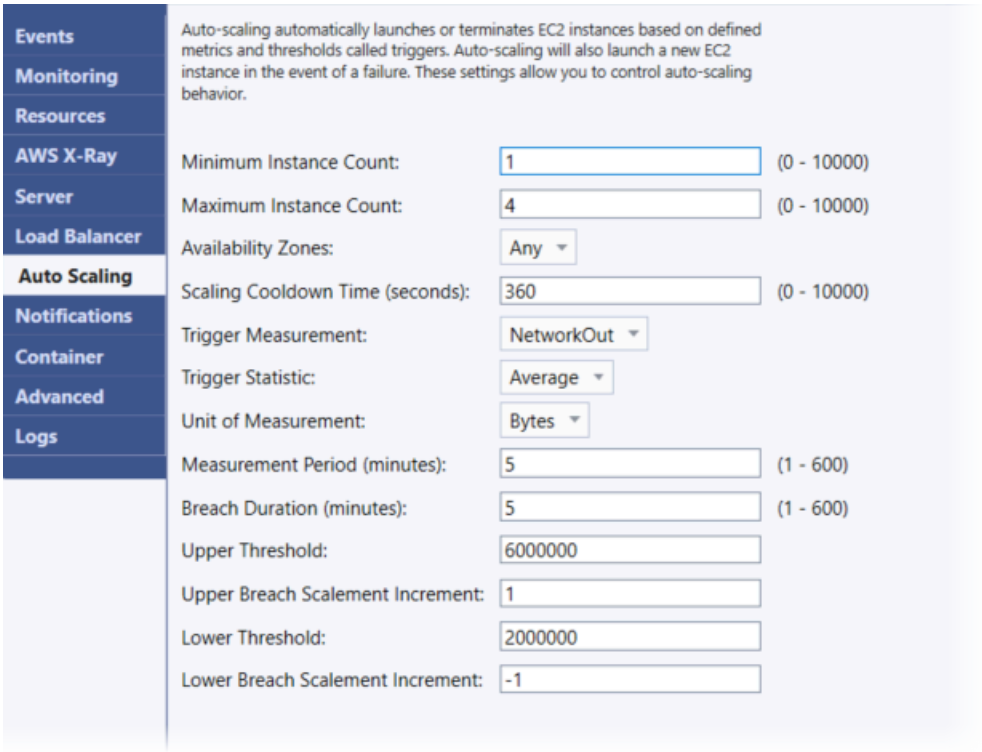
Amazon EC2 Auto Scaling 是一项 Amazon Web 服务，旨在根据用户定义的触发器自动启动或终止 Amazon EC2 实例。用户可以设置 Auto Scaling 组 并将触发器 与这些组关联，以根据带宽使用量或

CPU 利用率等指标自动扩展计算资源。Amazon EC2 Auto Scaling 与 Amazon CloudWatch 协作，检索运行应用程序的服务器实例的指标。

借助 Amazon EC2 Auto Scaling，您可以获取一组 Amazon EC2 实例并设置各种参数，使此组的数量自动增加或减少。Amazon EC2 Auto Scaling 可以在该组中添加或删除 Amazon EC2 实例，以帮助您无缝处理应用程序的流量变化。

Amazon EC2 Auto Scaling 还会监控其启动的每个 Amazon EC2 实例的运行状况。如果任何实例意外终止，Amazon EC2 Auto Scaling 会检测到终止情况并启动替换实例。这一功能可让您自动维护固定的、预期数量的 Amazon EC2 实例。

Elastic Beanstalk 为您的应用程序预配置 Amazon EC2 Auto Scaling。您可以使用 AWS Toolkit for Visual Studio 的应用程序环境选项卡内的 Auto Scaling 选项卡来编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

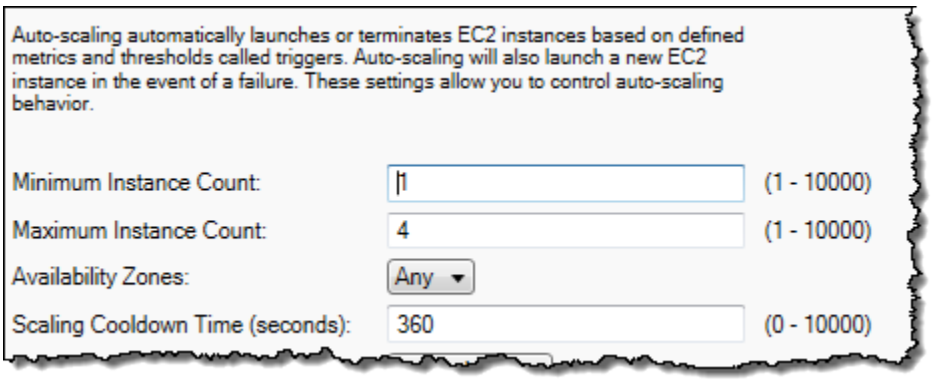


以下部分介绍了如何配置您的应用程序的 Auto Scaling 参数。

启动配置

您可以编辑启动配置以控制 Elastic Beanstalk 应用程序如何预配置 Amazon EC2 Auto Scaling 资源。

Minimum Instance Count（最小实例计数）和 Maximum Instance Count（最大实例计数）框可让您指定 Elastic Beanstalk 应用程序使用的 Auto Scaling 组的最小大小和最大大小。



> **Note**
>
> 要保持固定数量的 Amazon EC2 实例，请将 Minimum Instance Count（最小实例计数）和 Maximum Instance Count（最大实例计数）设置为相同的值。

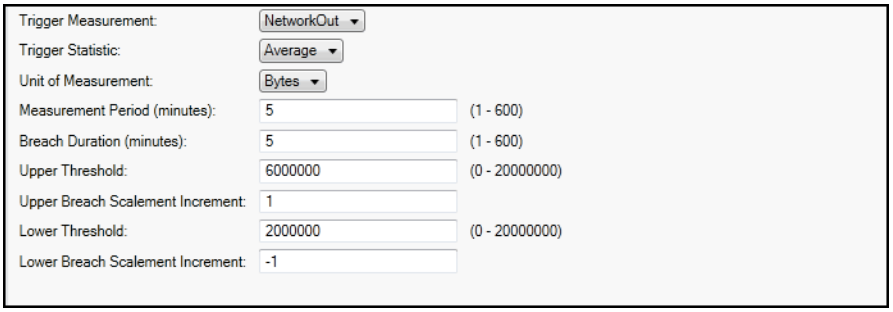
Availability Zones（可用区）框可让您指定希望 Amazon EC2 实例所在的可用区数。如果要构建容错的应用程序，则设置这个数字是十分重要的。如果一个可用区域出现故障，您的实例仍然会在其他可用区域中运行。

> **Note**
>
> 目前，您无法指定您的实例将放入哪些可用区域。

触发

触发器 是您设置的 Amazon EC2 Auto Scaling 机制，用于告知系统何时增加（扩展）和减少（缩减）实例数量。您可以配置这些触发器，使其在将任何指标（例如 CPU 利用率）发布到 Amazon CloudWatch 时激发，并确定是否已满足您指定的条件。当在指定的时间期限内超过为该指标指定的条件上限或者下限时，该触发会启动名为扩展活动 的长期运行流程。

您可以使用 AWS Toolkit for Visual Studio 为 Elastic Beanstalk 应用程序定义扩展触发器。



Amazon EC2 Auto Scaling 会通过监控特定实例的具体 Amazon CloudWatch 指标触发各种操作。指标包括 CPU 使用率、网络流量和磁盘活动。使用 Trigger Measurement（触发测量标准）设置选择触发的指标。

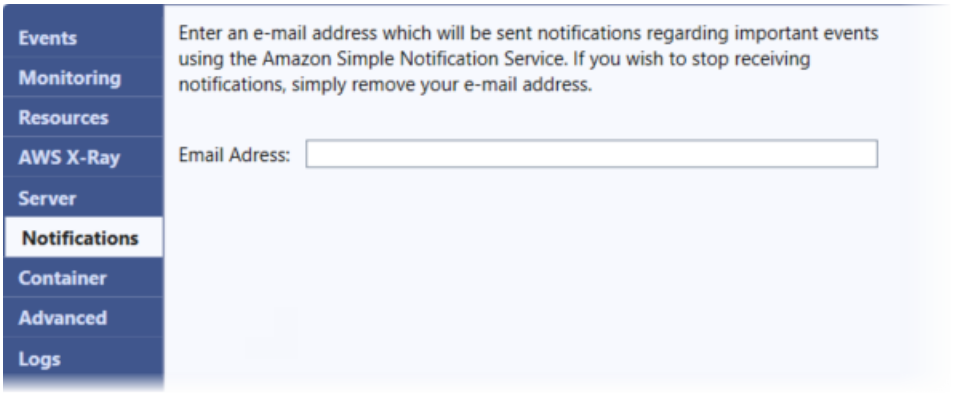
下表介绍了您可以使用 AWS 管理控制台配置的触发参数。

- 您可以指定该触发应该使用的统计数据。可以为 Trigger Statistic（触发统计数据）选择 Minimum（最小值）、Maximum（最大值）、Sum（总计）或 Average（平均值）。
- 对于 Unit of Measurement（测量单位），指定触发测量单位。
- 测量周期框内的值指定了 Amazon CloudWatch 对触发指标进行测量的频率。违例持续时间是在激活触发器之前，指标可以超出所定义的限制范围（如上限和下限所指定）的时长。
- 对于 Upper Breach Scale Increment（上限违例扩展增量）和 Lower Breach Scale Increment（下限违例扩展增量），指定执行扩展活动时要添加或删除的 Amazon EC2 实例数。

有关 Amazon EC2 Auto Scaling 的更多信息，请参阅 Amazon Elastic Compute Cloud 文档上的 Amazon EC2 Auto Scaling 部分。

使用 AWS Toolkit for Visual Studio 配置通知

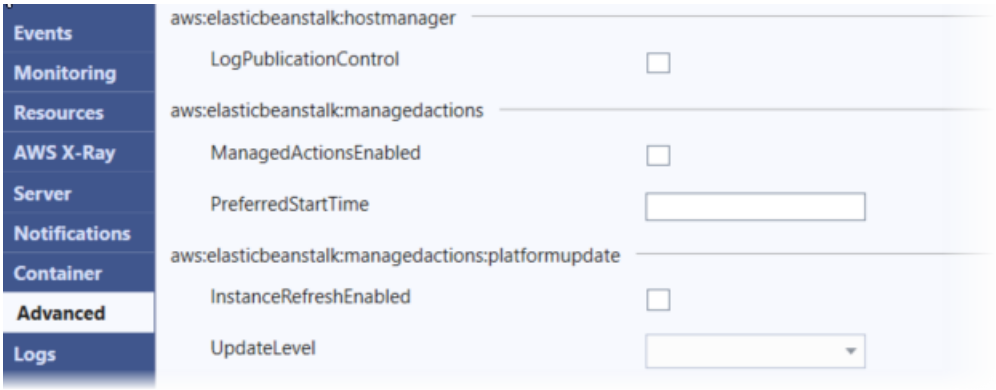
Elastic Beanstalk 使用 Amazon Simple Notification Service (Amazon SNS) 向您通知影响应用程序的重要事件。要启用 Amazon SNS 通知，请在 Email Address（电子邮件地址）框中输入您的电子邮件地址。若要禁用这些通知，请从框中删除您的电子邮件地址。



使用 AWS Toolkit for Visual Studio 配置其他环境选项

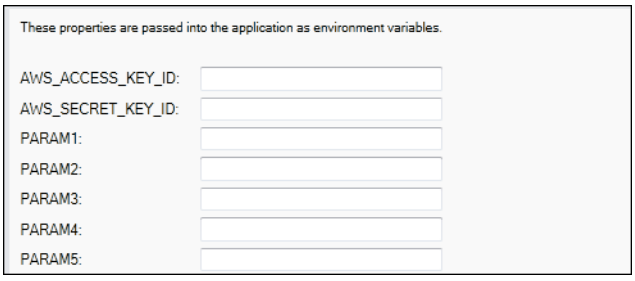
Elastic Beanstalk 定义了大量配置选项，您可以使用这些选项来配置环境的行为及其包含的资源。配置选项整理到类似于 `aws:autoscaling:asg` 的命名空间中。每个命名空间都定义了用于环境的 Auto Scaling 组的选项。高级面板按字母顺序列出了可在创建环境后更新的配置选项命名空间。

有关命名空间和选项的完整列表，包括各自的默认值和支持的值，请参阅面向所有环境的常规选项和Linux 上的 .NET Core 平台选项。



使用 AWS Toolkit for Visual Studio 配置 .NET Core 容器

容器面板可让您指定可从应用程序代码中读取的环境变量。



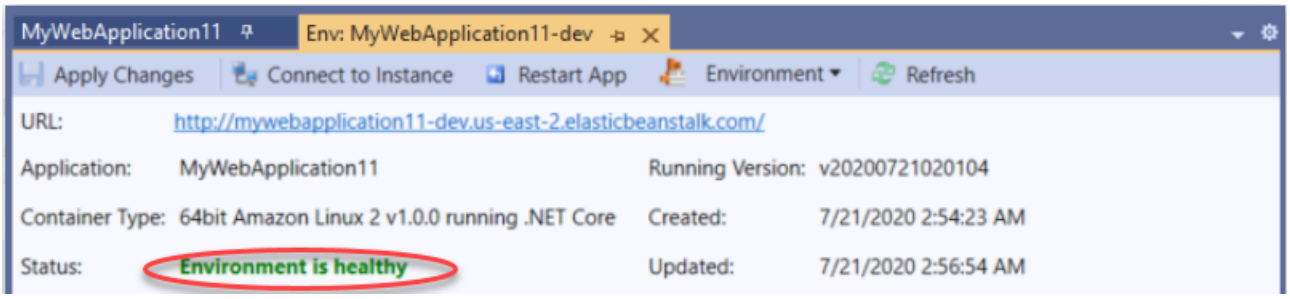
监控应用程序运行状况

了解您的生产网站是否可用以及能否对请求做出响应非常重要。Elastic Beanstalk 提供了功能来帮助您监控应用程序的响应能力。它监控有关应用程序的统计信息，并在超出阈值时向您发出警报。

有关 Elastic Beanstalk 提供的运行状况监控的信息，请参阅基本运行状况报告。

您可以使用 AWS Toolkit for Visual Studio 或者 AWS 管理控制台访问关于应用程序的操作信息。

该工具包会在状态字段中显示您环境的状态和应用程序运行状况。



监控应用程序运行状况

1. 在 AWS Toolkit for Visual Studio 的 AWS Explorer 中，展开 Elastic Beanstalk 节点，然后展开您的应用程序节点。
2. 打开应用程序环境的上下文（右键单击）菜单，然后选择查看状态。
3. 在应用程序环境选项卡上，选择监控。

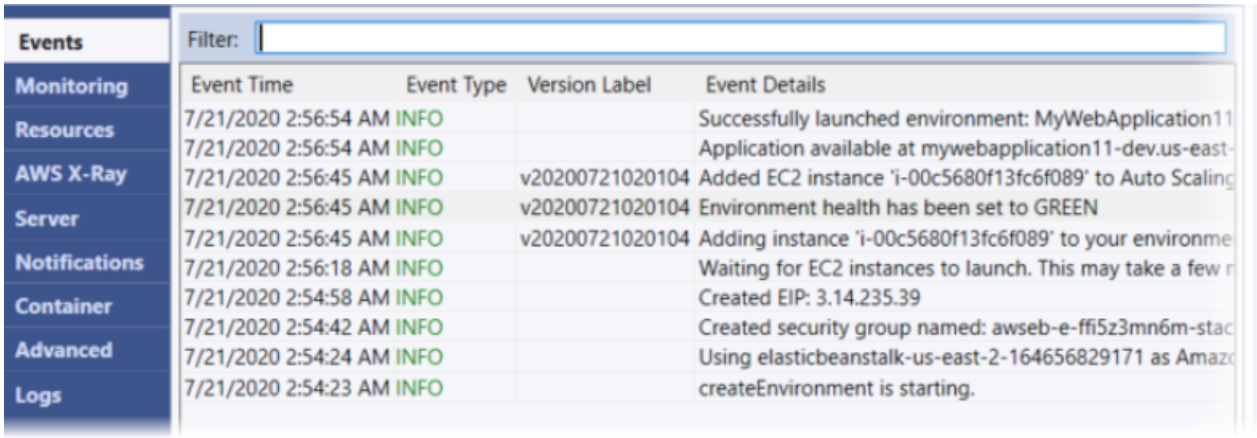
   监控面板包含一组图表，用于显示您特定应用程序环境的资源使用情况。

> **Note**
>
> 默认情况下，时间范围设置为最近一个小时。要修改此设置，请在时间范围列表中选择另一个时间范围。

您可以使用 AWS Toolkit for Visual Studio 或者 AWS 管理控制台，查看与您的应用程序相关联的事件。

查看应用程序事件

1. 在 AWS Toolkit for Visual Studio 的 AWS Explorer 中，展开 Elastic Beanstalk 节点和您的应用程序节点。
2. 打开应用程序环境的上下文（右键单击）菜单，然后选择查看状态。
3. 在应用程序环境选项卡上，选择事件。



从 .NET on Windows Server 平台迁移到 .NET Core on Linux 平台

您可以将在 .NET on Windows Server 平台上运行的应用程序迁移到 .NET Core on Linux 平台。以下是从 Windows 平台迁移到 Linux 平台时的一些注意事项。

迁移到 .NET Core on Linux 平台的注意事项

领域	更改和信息
应用程序配置	在 Windows 平台上，您可以使用部署清单来指定在您的环境中运行的应用程序。这些 .NET Core on Linux 平台使用 Procfile 指定在环境实例上运行的应用程序。有关捆绑应用程序的详细信息，请参阅the section called “捆绑应用程序”。
代理服务器	在 Windows 平台上，您可以使用 IIS 作为应用程序的代理服务器。默认情况下，.NET Core on Linux 平台使用 nginx 作为反向代理。您可以选择不使用代理服务器，并使用 Kestrel 作为应用程序的 Web 服务器。要了解更多信息，请参阅“the section called “代理服务器””。
路由选择	在 Windows 平台上，您可以在应用程序代码中使用 IIS，并添加部署清单来配置 IIS 路径。对于 .NET Core on Linux 平台，您可以在应用程序代码中使用 ASP .NET Core 路由，并更新环境的 nginx 配置。要了解更多信息，请参阅“the section called “代理服务器””。
日志	Linux 和 Windows 平台会流式传输不同的日志。有关详细信息，请参阅the section called “Elastic Beanstalk 如何设置 CloudWatch Logs”。

在 Elastic Beanstalk 上创建和部署.NET Windows 应用程序

> **在 AWS 开发者中心查看.NET**
>
> 你来过我们的 .Net 开发者中心了吗？ 这是我们提供.NET 上所有内容的一站式商店 AWS。有关更多信息，请参阅AWS 开发人员中心上的.NET。

AWS Elastic Beanstalk for .NET 可以更轻松地部署、管理和扩展使用 Amazon Web Services 的 ASP.NET 和.NET Core Web 应用程序。本章提供有关创建、测试、部署您的 Windows Web 应用程序以及将其重新部署到 Elastic Beanstalk 的说明。您可以使用 Elastic Beanstalk 命令行界面 (EB CLI) 或使用 Elastic Beanstalk 控制台在短短几分钟内部署应用程序。

本章提供以下教程：

- QuickStart 适用于 Windows 上的.NET 核心版

- 部署 ASP.NET 核心应用程序

如果你在 Windows .NET Core 应用程序开发方面需要帮助，可以去以下几个地方：

- .NET 开发论坛 — 发布您的问题并获得反馈。
- .NET 开发人员中心 — 提供示例代码、文档、工具和其他资源的一站式商店。
- AWS 适用于.NET 的 SDK 文档 — 阅读有关设置 SDK 和运行代码示例、SDK 功能以及有关 SDK 的 API 操作的详细信息。

> **Note**
>
> 此平台不支持以下 Elastic Beanstalk 功能：
>
> - 工作线程环境。有关详细信息，请参阅Elastic Beanstalk 工作线程环境。
> - 捆绑日志。有关详细信息，请参阅查看实例日志。

主题

已停用的 Elastic Beanstalk Windows 2012 平台分支和 TLS 1.2 兼容性

如果您的应用程序当前正在停用的 Windows Server 2012 R2 平台分支上运行，则本主题提供了建议。它还解决了我们的 AWS 服务 API 端点和受影响的平台分支上已弃用的 TLS 1.0 和 1.1 协议版本支持。

Windows Server 2012 R2 平台分支已停用

Elastic Beanstalk 于 2023 年 12 月 4 日停用了 Windows Server 2012 R2 平台分支，并于 2024 年 4 月 10 日将与这些平台关联的 AMI 私有化。此操作可防止在你的 Windows Server 2012 环境中启动使用默认 Beanstalk AMI 的实例。

如果您有任何环境在停用的 Windows 平台分支上运行，我们建议您将它们迁移到以下 Windows Server 平台之一，这些平台是最新且完全受支持的：

- 带有 IIS 10.0 版本 2.x 的 Windows 服务器 2022
- Windows Server 2019 with IIS 10.0 版本 2.x

有关完整迁移注意事项，请参阅 从 Windows Server 平台更早的主要版本迁移。

有关平台弃用的更多信息，请参阅 Elastic Beanstalk 平台支持策略。

> **Note**
>
> 如果你无法迁移到这些完全支持的平台，我们建议你使用使用 Windows Server 2012 R2 或 Windows Server 2012 R2 Core AMI 创建的自定义 AMI 作为基础映像（如果你还没有这样做）。有关详细说明，请参阅 保持能够访问适用于已停用平台的亚马逊机器映像（AMI）的方法。如果您在执行其中一个迁移步骤时需要临时访问 AMI，请联系 Su AWS pport Center。

TLS 1.2 兼容性

自 2023 年 12 月 31 日 AWS 起，已开始在所有 AWS API 端点上全面实施 TLS 1.2。此操作取消了在所有 AWS API 中使用 TLS 版本 1.0 和 1.1 的功能。该信息最初是在 2022年6月28日发布的。要避免影响可用性的风险，请尽快将运行此处标识的平台版本的所有环境升级到新版本（如果您尚未这样做）。

潜在影响

运行 TLS v1.1 或更早版本的 Elastic Beanstalk 平台版本会受到影响。此更改会影响包括但不限于以下内容的环境操作：配置部署、应用程序部署、自动扩展、新环境启动、日志轮换、增强运行状况报告以及将应用程序日志发布到与您的应用程序关联的 Amazon S3 存储桶。

受影响的 Windows 平台版本

建议在以下平台版本上使用 Elastic Beanstalk 环境的客户将每个相应环境升级到 2022 年 2 月 18 日发布的 Windows 平台版本 2.8.3 或更高版本。

- Windows Server 2019：平台版本 2.8.2 或先前版本

建议在以下平台版本上使用 Elastic Beanstalk 环境的客户将其每个相应环境升级到 2022 年 12 月 28 日发布的 Windows 平台版本 2.10.7 或更高版本。

- Windows Server 2016 — 平台版本 2.10.6 或之前的版本
- Windows Server 2012 — 所有平台版本；该平台已于 2023 年 12 月 4 日停用
- Windows Server 2008 — 所有平台版本；该平台已于 2019 年 10 月 28 日停用

有关最新的受支持 Windows Server 平台版本列表，请参阅《AWS Elastic Beanstalk 平台指南》中的支持的平台。

有关更新环境的详细信息和最佳实践，请参阅 更新 Elastic Beanstalk 环境的平台版本。

QuickStart: 在 Windows 应用程序上部署.NET 核心到 Elastic Beanstalk

本 QuickStart 教程将引导你完成在 Windows 应用程序上创建.NET Core 并将其部署到 AWS Elastic Beanstalk 环境的过程。

> **Note**
>
> 本 QuickStart 教程仅用于演示目的。不要将本教程中创建的应用程序用于生产流量。

Sections

- 后续步骤
- 使用 Elastic Beanstalk 控制台进行部署

你的 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

如果您已经有一个 AWS 帐户，则可以继续前进先决条件 。

创建一个 AWS 账户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

先决条件

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，前面有提示符号 (>) 和当前目录的名称（如果适用）。

```
C:\eb-project> this is a command
this is output
```

EB CLI

本教程使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

Windows 上的.NET 内核

如果您的本地计算机上没有安装.NET SDK，则可以通过.NET 文档网站上的 “下载.NE T” 链接进行安装。

运行以下命令验证您的.NET SDK 安装情况。

```
C:\> dotnet --info
```

步骤 1：在 Windows 应用程序上创建.NET 内核

创建项目目录。

```
C:\> mkdir eb-dotnetcore
C:\> cd eb-dotnetcore
```

接下来，通过运行以下命令创建示例 Hello World RESTful Web 服务应用程序。

```
C:\eb-dotnetcore> dotnet new web --name HelloElasticBeanstalk
C:\eb-dotnetcore> cd HelloElasticBeanstalk
```

步骤 2：在本地运行应用程序

运行以下命令以在本地运行应用程序。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> dotnet run
```

输出应类似于以下文本。

```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7222
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5228
info: Microsoft.Hosting.Lifetime[0]
```

```
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Administrator\eb-dotnetcore\HelloElasticBeanstalk
```

> **Note**
>
> 在本地运行应用程序时，该`dotnet`命令会随机选择一个端口。在本示例中，端口为 5228。当您将应用程序部署到 Elastic Beanstalk 环境时，该应用程序将在端口 5000 上运行。

`http://localhost:`*`port`*在您的网络浏览器中输入 URL 地址。对于这个具体的示例，命令是`http://localhost:5228`。网络浏览器应显示 “Hello World！”。

第 3 步：使用 EB CLI 在 Windows 应用程序上部署.NET 核心

运行以下命令为此应用程序创建 Elastic Beanstalk 环境。

创建环境并在 Windows 应用程序上部署您的.NET Core

1. 在`HelloElasticBeanstalk`目录中运行以下命令以发布和压缩应用程序。

   ```
   C:\eb-dotnetcore\HelloElasticBeasntalk> dotnet publish -o site
   C:\eb-dotnetcore\HelloElasticBeasntalk> cd site
   C:\eb-dotnetcore\HelloElasticBeasntalk\site> Compress-Archive -Path * -
   DestinationPath ../site.zip
   C:\eb-dotnetcore\HelloElasticBeasntalk\site> cd ..
   ```

2. 在`HelloElasticBeanstalk`调用的中创建一个`aws-windows-deployment-manifest.json`包含以下内容的新文件：

   ```
   {
       "manifestVersion": 1,
       "deployments": {
           "aspNetCoreWeb": [
           {
               "name": "test-dotnet-core",
               "parameters": {
                   "appBundle": "site.zip",
                   "iisPath": "/",
   ```

```
                "iisWebSite": "Default Web Site"
            }
        }
        ]
    }
}
```

3. 使用 eb init 命令，初始化 EB CLI 存储库。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb init -p iis dotnet-windows-server-
tutorial --region us-east-2
```

此命令创建名为的应用程序，dotnet-windows-server-tutorial并将您的本地存储库配置为使用最新 Windows 服务器平台版本创建环境。

4. 创建环境并使用 eb create 将应用程序部署到此环境中。Elastic Beanstalk 会自动为您的应用程序生成一个 zip 文件并在端口 5000 上启动该文件。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb create dotnet-windows-server-env
```

Elastic Beanstalk 创建您的环境大约需要五分钟。

第 4 步：在 Elastic Beanstalk 上运行你的应用程序

创建环境的过程完成后，使用打开您的网站eb open。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb open
```

恭喜您！你已经使用 Elastic Beanstalk 在 Windows 应用程序上部署了.NET 核心！这将使用为应用程序创建的域名打开一个浏览器窗口。

第 5 步：清理

完成应用程序的使用后，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源。

要使用 EB CLI 终止您的 Elastic Beanstalk 环境，请运行以下命令。

```
C:\eb-dotnetcore\HelloElasticBeasntalk> eb terminate
```

AWS 您的应用程序的资源

您刚刚创建了一个单实例应用程序。它可用作带有单个 EC2 实例的简单示例应用程序，因此不需要负载平衡或 auto Scaling。对于单实例应用程序，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

  各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 管理所有这些资源。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。您还可以使用 Elastic Beanstalk 控制台探索您的新环境。有关详细步骤，请参阅本指南入门一章中的探索您的环境。

部署一两个示例应用程序并准备好开始在 Windows 应用程序上本地开发和运行.NET Core 之后，请参阅 设置 .NET 开发环境

使用 Elastic Beanstalk 控制台进行部署

您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。

教程：使用 Elastic Beanstalk 部署 ASP.NET Core 应用程序

在本教程中，您将逐步完成构建新的 ASP.NET Core 应用程序并将其部署到 AWS Elastic Beanstalk的过程。

首先，您将使用 .NET Core 开发工具包的 `dotnet` 命令行工具生成基本命令行 .NET Core 应用程序、安装依赖项、编译代码并在本地运行应用程序。接下来，您将创建默认 `Program.cs` 类，并添加 ASP.NET `Startup.cs` 类和配置文件，以创建使用 ASP.NET 和 IIS 为 HTTP 请求提供服务的应用程序。

最后，Elastic Beanstalk 使用部署清单在一个服务器上配置 .NET Core 应用程序、自定义应用程序和多个 .NET Core 或 MSBuild 应用程序的部署。要将 .NET Core 应用程序部署到 Windows Server 环境，请使用部署清单将站点存档添加到应用程序源包。`dotnet publish` 命令生成已编译的类和依赖项，您可以将它们与 `web.config` 文件捆绑，以创建一个站点存档。部署清单告知 Elastic Beanstalk 站点应运行的路径且可用于配置应用程序池并在不同路径上运行多个应用程序。

源代码可在此处获得：dotnet-core-windows-tutorial.zip

Sections

先决条件

本教程使用 .NET Core 开发工具包生成基本 .NET Core 应用程序，在本地运行该应用程序，并构建可部署的软件包。

要求

- .NET Core (x64) 1.0.1、2.0.0 或更高版本

安装 .NET Core 开发工具包

1. 从 microsoft.com/net/core 下载安装程序。选择 Windows。选择 Download .NET SDK（下载 .NET 开发工具包）。
2. 运行安装程序并按说明操作。

本教程使用命令行 ZIP 实用工具创建一个源包，您可以将该源包部署到 Elastic Beanstalk。要在 Windows 中使用 `zip` 命令，您可以安装 `UnxUtils`，一个有用的命令行实用工具 (如 `zip` 和 `ls`) 的轻型集合。或者，可以使用 Windows 资源管理器或任何其他 ZIP 实用工具创建源包存档。

要安装 UnxUtils

1. 下载 `UnxUtils`。
2. 将存档提取到本地目录。例如：`C:\Program Files (x86)`。
3. 将二进制的路径添加到 Windows PATH 用户变量。例如：`C:\Program Files (x86)\UnxUtils\usr\local\wbin`。
   a. 按下 Windows 键，然后键入 **environment variables**。
   b. 选择 Edit environment variables for your account（编辑您账户的环境变量）。
   c. 选择 PATH，然后选择 Edit（编辑）。
   d. 向 Variable value 字段添加路径，中间用分号隔开。例如：*C:\item1\path*;***C:\item2\path***
   e. 选择 OK 两次以应用新设置。
   f. 关闭任何正在运行的命令提示符窗口，然后重新打开命令提示符窗口。
4. 打开一个新的命令提示符窗口并运行 `zip` 命令，验证它是否有效。

```
> zip -h
Copyright (C) 1990-1999 Info-ZIP
Type 'zip "-L"' for software license.
...
```

生成 .NET Core 项目

使用 `dotnet` 命令行工具生成一个新的 C# .NET Core 项目并在本地运行它。默认 .NET Core 应用程序是打印 `Hello World!` 然后退出的命令行实用工具。

生成新的 .NET Core 项目

1. 打开一个新的命令提示窗口并导航至您的用户文件夹。

```
> cd %USERPROFILE%
```

2. 使用 dotnet new 命令生成新的 .NET Core 项目。

```
C:\Users\username> dotnet new console -o dotnet-core-tutorial
Content generation time: 65.0152 ms
The template "Console Application" created successfully.
C:\Users\username> cd dotnet-core-tutorial
```

3. 使用 dotnet restore 命令安装依赖项。

```
C:\Users\username\dotnet-core-tutorial> dotnet restore
Restoring packages for C:\Users\username\dotnet-core-tutorial\dotnet-core-
tutorial.csproj...
Generating MSBuild file C:\Users\username\dotnet-core-tutorial\obj\dotnet-core-
tutorial.csproj.nuget.g.props.
Generating MSBuild file C:\Users\username\dotnet-core-tutorial\obj\dotnet-core-
tutorial.csproj.nuget.g.targets.
Writing lock file to disk. Path: C:\Users\username\dotnet-core-tutorial\obj
\project.assets.json
Restore completed in 1.25 sec for C:\Users\username\dotnet-core-tutorial\dotnet-
core-tutorial.csproj.

NuGet Config files used:
    C:\Users\username\AppData\Roaming\NuGet\NuGet.Config
    C:\Program Files (x86)\NuGet\Config\Microsoft.VisualStudio.Offline.config
Feeds used:
    https://api.nuget.org/v3/index.json
    C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\
```

4. 使用 dotnet run 命令在本地构建和运行应用程序。

```
C:\Users\username\dotnet-core-tutorial> dotnet run
Hello World!
```

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台启动 Elastic Beanstalk 环境。对于本示例，您将同时启动 .NET 平台。启动和配置环境后，可随时部署新的源代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &Environmentype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建需要花费大约 10 分钟。再次期间，您可以更新源代码。

更新源代码

将默认应用程序修改为使用 ASP.NET 和 IIS 的 Web 应用程序。

- ASP.NET 是 .NET 的网站框架。
- IIS 是在 Elastic Beanstalk 环境中的 Amazon EC2 实例上运行应用程序的 Web 服务器。

以下是要遵循的源代码示例：dotnet-core-tutorial-source. zip

> **Note**
>
> 以下过程介绍如何将项目代码转换为 Web 应用程序。为简化该过程，您可以从头开始就将此项目生成为一个 Web 应用程序。在上一节生成 .NET Core 项目中，使用以下命令修改 `dotnet new` 步骤的命令。
>
> ```
> C:\Users\username> dotnet new web -o dotnet-core-tutorial -n WindowsSampleApp
> ```

将 ASP.NET 和 IIS 支持添加到您的代码

1. 将 `Program.cs` 复制到应用程序目录中以作为 Web 主机生成器运行。

Example c:\users\username\dotnet-core-tutorial\ Program.cs

```
namespace Microsoft.AspNetCore.Hosting;
using WindowsSampleApp;

public static class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args).UseStartup<Startup>();
}
```

2. 添加 `Startup.cs` 以运行 ASP.NET 网站。

   Example c:\users\username\dotnet-core-tutorial\ Startup.cs

   ```
   namespace WindowsSampleApp
   {
       public class Startup
       {
           public void Configure(IApplicationBuilder app)
           {
               app.UseRouting();
               app.UseEndpoints(endpoints =>
               {
                   endpoints.MapGet("/", () => "Hello World from Elastic Beanstalk");
               });
           }
       }
   }
   ```

3. 添加 `WindowsSampleApp.csproj`，其中包含 IIS 中间件，以及来自 `web.config` 的输出的 `dotnet publish` 文件。

> **Note**
>
> 以下示例是使用 .NET Core 运行时 2.2.1 开发的。您可能需要修改 `TargetFramework` 元素中的 `Version` 或 `PackageReference` 属性值以匹配您在自定义项目中使用的 .NET Core 运行时版本。

Example c:\users\username\dotnet-core-tutorial\ WindowsSampleApp .csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <RollForward>LatestMajor</RollForward>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>WindowsSampleApp</RootNamespace>
  </PropertyGroup>

</Project>
```

接下来，安装新的依赖项并在本地运行 ASP.NET 网站。

在本地运行网站

1. 使用 `dotnet restore` 命令安装依赖项。
2. 使用 `dotnet run` 命令在本地构建和运行应用程序。
3. 打开 localhost:5000 查看网站。

要在 Web 服务器上运行应用程序，您需要将已编译的源代码与 `web.config` 配置文件和运行时依赖项捆绑。`dotnet` 工具提供了一个 `publish` 命令，该命令根据 `dotnet-core-tutorial.csproj` 中的配置将这些文件聚集在一个目录中。

构建您的网站

- 使用 `dotnet publish` 命令将已编译代码和依赖项输出到一个名为 `site` 的文件夹。

```
C:\users\username\dotnet-core-tutorial> dotnet publish -o site
```

要将应用程序部署到 Elastic Beanstalk，请将站点存档与部署清单捆绑。这将告知 Elastic Beanstalk 如何运行它。

创建源包

1. 将站点文件夹中的文件添加到 ZIP 文档。

> **Note**
>
> 如果您使用不同的 ZIP 实用程序，请确保将所有文件添加到生成的 ZIP 存档的根文件夹中。这是成功将应用程序部署到您的 Elastic Beanstalk 环境所必需的。

```
C:\users\username\dotnet-core-tutorial> cd site
C:\users\username\dotnet-core-tutorial\site> zip ../site.zip *
  adding: dotnet-core-tutorial.deps.json (164 bytes security) (deflated 84%)
  adding: dotnet-core-tutorial.dll (164 bytes security) (deflated 59%)
  adding: dotnet-core-tutorial.pdb (164 bytes security) (deflated 28%)
  adding: dotnet-core-tutorial.runtimeconfig.json (164 bytes security) (deflated
 26%)
  adding: Microsoft.AspNetCore.Authentication.Abstractions.dll (164 bytes security)
 (deflated 49%)
  adding: Microsoft.AspNetCore.Authentication.Core.dll (164 bytes security)
 (deflated 57%)
  adding: Microsoft.AspNetCore.Connections.Abstractions.dll (164 bytes security)
 (deflated 51%)
  adding: Microsoft.AspNetCore.Hosting.Abstractions.dll (164 bytes security)
 (deflated 49%)
  adding: Microsoft.AspNetCore.Hosting.dll (164 bytes security) (deflated 60%)
  adding: Microsoft.AspNetCore.Hosting.Server.Abstractions.dll (164 bytes security)
 (deflated 44%)
  adding: Microsoft.AspNetCore.Http.Abstractions.dll (164 bytes security) (deflated
 54%)
  adding: Microsoft.AspNetCore.Http.dll (164 bytes security) (deflated 55%)
  adding: Microsoft.AspNetCore.Http.Extensions.dll (164 bytes security) (deflated
 50%)
  adding: Microsoft.AspNetCore.Http.Features.dll (164 bytes security) (deflated
 50%)
```

```
 adding: Microsoft.AspNetCore.HttpOverrides.dll (164 bytes security) (deflated
49%)
 adding: Microsoft.AspNetCore.Server.IISIntegration.dll (164 bytes security)
(deflated 46%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Core.dll (164 bytes security)
(deflated 63%)
 adding: Microsoft.AspNetCore.Server.Kestrel.dll (164 bytes security) (deflated
46%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Https.dll (164 bytes security)
(deflated 44%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Transport.Abstractions.dll (164 bytes
security) (deflated 56%)
 adding: Microsoft.AspNetCore.Server.Kestrel.Transport.Sockets.dll (164 bytes
security) (deflated 51%)
 adding: Microsoft.AspNetCore.WebUtilities.dll (164 bytes security) (deflated 55%)
 adding: Microsoft.Extensions.Configuration.Abstractions.dll (164 bytes security)
(deflated 48%)
 adding: Microsoft.Extensions.Configuration.Binder.dll (164 bytes security)
(deflated 47%)
 adding: Microsoft.Extensions.Configuration.dll (164 bytes security) (deflated
46%)
 adding: Microsoft.Extensions.Configuration.EnvironmentVariables.dll (164 bytes
security) (deflated 46%)
 adding: Microsoft.Extensions.Configuration.FileExtensions.dll (164 bytes
security) (deflated 47%)
 adding: Microsoft.Extensions.DependencyInjection.Abstractions.dll (164 bytes
security) (deflated 54%)
 adding: Microsoft.Extensions.DependencyInjection.dll (164 bytes security)
(deflated 53%)
 adding: Microsoft.Extensions.FileProviders.Abstractions.dll (164 bytes security)
(deflated 46%)
 adding: Microsoft.Extensions.FileProviders.Physical.dll (164 bytes security)
(deflated 47%)
 adding: Microsoft.Extensions.FileSystemGlobbing.dll (164 bytes security)
(deflated 49%)
 adding: Microsoft.Extensions.Hosting.Abstractions.dll (164 bytes security)
(deflated 47%)
 adding: Microsoft.Extensions.Logging.Abstractions.dll (164 bytes security)
(deflated 54%)
 adding: Microsoft.Extensions.Logging.dll (164 bytes security) (deflated 48%)
 adding: Microsoft.Extensions.ObjectPool.dll (164 bytes security) (deflated 45%)
 adding: Microsoft.Extensions.Options.dll (164 bytes security) (deflated 53%)
 adding: Microsoft.Extensions.Primitives.dll (164 bytes security) (deflated 50%)
 adding: Microsoft.Net.Http.Headers.dll (164 bytes security) (deflated 53%)
```

```
  adding: System.IO.Pipelines.dll (164 bytes security) (deflated 50%)
  adding: System.Runtime.CompilerServices.Unsafe.dll (164 bytes security) (deflated
 43%)
  adding: System.Text.Encodings.Web.dll (164 bytes security) (deflated 57%)
  adding: web.config (164 bytes security) (deflated 39%)
C:\users\username\dotnet-core-tutorial\site> cd ../
```

2. 添加指向站点存档的部署清单。

   Example c:\users\username\dotnet-core-tutorial\ aws-windows-deployment-manifest .json

```
{
    "manifestVersion": 1,
    "deployments": {
        "aspNetCoreWeb": [
        {
            "name": "test-dotnet-core",
            "parameters": {
                "appBundle": "site.zip",
                "iisPath": "/",
                "iisWebSite": "Default Web Site"
            }
        }
        ]
    }
}
```

3. 使用 `zip` 命令创建一个名为 `dotnet-core-tutorial.zip` 的源包。

```
C:\users\username\dotnet-core-tutorial> zip dotnet-core-tutorial.zip site.zip aws-
windows-deployment-manifest.json
  adding: site.zip (164 bytes security) (stored 0%)
  adding: aws-windows-deployment-manifest.json (164 bytes security) (deflated 50%)
```

部署您的应用程序

将源包部署到您创建的 Elastic Beanstalk 环境。

你可以在这里下载源包：dotnet-core-tutorial-bundle. zip

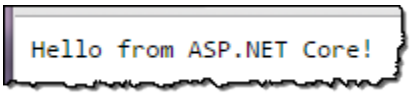
部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

应用程序简单地将 `Hello from ASP.NET Core!` 写入到响应并返回。



启动环境将创建以下资源：

- EC2 实例 - 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud (Amazon EC2) 虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。

- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

后续步骤

当您继续开发应用程序时，您可能希望管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

如果您使用 Visual Studio 开发应用程序，则还可以使用 AWS Toolkit for Visual Studio 来部署变更、管理 Elastic Beanstalk 环境以及管理其他资源。 AWS 参阅 这些区域有： AWS Toolkit for Visual Studio 了解更多信息。

对于开发和测试，您可能希望使用 Elastic Beanstalk 的用于将托管数据库实例直接添加到环境的功能。有关在环境内设置数据库的说明，请参阅将数据库添加到 Elastic Beanstalk 环境。

最后，如果您计划在生产环境中使用应用程序，请为环境配置自定义域名并启用 HTTPS 以实现安全连接。

设置 .NET 开发环境

设置 .NET 开发环境以在本地测试应用程序，然后再将它部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

有关适用于所有语言的常见设置步骤和工具，请参阅配置用于 Elastic Beanstalk 的开发计算机。

小节目录

如果您需要在应用程序中管理 AWS 资源，请安装 AWS SDK for .NET。例如，您可以使用 Amazon S3 存储和检索数据。

通过 AWS SDK for .NET，您可以在几分钟内通过可下载的“一站式”包开始使用，其中包含了全部的 Visual Studio 项目模板、AWS NET 库、C# 代码示例和文档。对于如何使用库构建应用程序，C# 也提供了很多实用的示例。我们提供了在线视频教程和参考文档，以帮助您了解如何使用库和代码示例。

请访问 AWS SDK for .NET 主页以了解更多信息和安装说明。

安装 IDE

集成开发环境 (IDE) 提供了便于应用程序开发的大量功能。如果您尚未使用 IDE 进行 .NET 开发，请尝试 Visual Studio Community 来开始操作。

请访问 Visual Studio Community 页以下载并安装 Visual Studio Community。

安装 AWS Toolkit for Visual Studio

AWS Toolkit for Visual Studio 是适用于 Visual Studio IDE 的开源插件，能够让开发人员更轻松地使用 AWS 开发、调试和部署 .NET 应用程序。有关安装说明，请访问 Toolkit for Visual Studio 主页。

使用 Elastic Beanstalk .NET 平台

AWS Elastic Beanstalk 支持适用于不同版本的.NET 编程框架和 Windows 服务器的多种平台。有关完整列表，请参阅 AWS Elastic Beanstalk 平台文档中的使用 IIS 的 Windows Server 上的 .NET。

Elastic Beanstalk 提供了配置选项，可供您用于自定义在 Elastic Beanstalk 环境中的 EC2 实例上运行的软件。您可以配置应用程序所需的环境变量，启用到 Amazon S3 的日志轮换，并设定 .NET Framework 设置。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

在 Elastic Beanstalk 控制台中配置 .NET 环境

您可以使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，配置应用程序可以从环境中读取的变量以及更改 .NET Framework 设置。

在 Elastic Beanstalk 控制台中配置 .NET 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

容器选项

- Target .NET runtime（目标 .NET 运行时）– 设置为 `2.0` 以运行 CLR v2。
- Enable 32-bit applications（启用 32 位应用程序）– 设置为 `True` 可运行 32 位应用程序。

日志选项

“日志选项”部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

环境属性

在环境属性部分，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。这些设置会以密钥值对的方式传递到应用程序。使用 `System.GetEnvironmentVariable` 可读取它们。相同的密钥可以存在于 `web.config` 中，也可以作为环境属性存在。使用 `System.Configuration` 命名空间可读取 `web.config` 中的值。

```
NameValueCollection appConfig = ConfigurationManager.AppSettings;
string endpoint = appConfig["API_ENDPOINT"];
```

参阅 环境属性和其他软件设置 了解更多信息。

aws:elasticbeanstalk:container:dotnet:apppool 命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

.NET 平台在 `aws:elasticbeanstalk:container:dotnet:apppool` 命名空间中定义可用来配置 .NET 运行时的选项。

以下示例配置文件显示这个命名空间中可用的每个选项的设置：

Example .ebextensions/dotnet-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:dotnet:apppool:
    Target Runtime: 2.0
    Enable 32-bit Applications: True
```

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI来配置选项。参阅 配置选项 了解更多信息。

跨 Elastic BeanstalkWindows Server 平台的主要版本迁移

AWS Elastic Beanstalk 其 Windows 服务器平台有多个主要版本。此页面介绍每个主要版本的主要改进以及在迁移到较新版本之前应考虑的事项。

Windows Server 平台当前处于版本 2 (v2)。如果应用程序使用 v2 之前的任何 Windows Server 平台版本，我们建议迁移到 v2。

Windows Server 平台的主要版本中的新增功能

Windows Server 平台 V2

Elastic Beanstalk Windows Server 平台的版本 2 (v2) 已于 2019 年 2 月发布。V2 使 Windows Server 平台的行为在多个重要方面更接近 Elastic Beanstalk 基于 Linux 的平台的行为。V2 完全向后兼容 v1，这使从 v1 迁移轻松。

Windows Server 平台现在支持以下功能：

- 版本控制 - 每个版本将获取新的版本号，并且在创建和管理环境时可以引用先前的版本（仍可用的版本）。
- 增强型运行状况 - 有关详细信息，请参阅增强型运行状况报告和监控。

- 不可变 和附加批次滚动 部署 - 有关部署策略的详细信息，请参阅将应用程序部署到 Elastic Beanstalk 环境。
- 不可变更新 - 有关更新类型的详细信息，请参阅配置更改。
- 托管平台更新 - 有关详细信息，请参阅托管平台更新。

> **Note**
>
> 新的部署和更新功能依赖于增强型运行状况。启用增强型运行状况以使用这些功能。有关详细信息，请参阅启用 Elastic Beanstalk 增强型运行状况报告。

Windows Server 平台 V1

Elastic Beanstalk Windows Server 平台的版本 1.0.0 (v1) 已于 2015 年 10 月发布。此版本改变了环境创建和更新期间 Elastic Beanstalk 处理配置文件中的命令的顺序。

之前的平台版本在解决方案堆栈名称中没有版本号：

- 运行 IIS 8.5 的 64 位 Windows Server 2012 R2
- 运行 IIS 8.5 的 64 位 Windows Server Core 2012 R2
- 运行 IIS 8 的 64 位 Windows Server 2012
- 运行 IIS 7.5 的 64 位 Windows Server 2008 R2

在之前的版本中，配置文件的处理顺序不一致。创建环境期间，`Container Commands` 在应用程序源部署到 IIS 后运行。在向正在运行的环境部署期间，容器命令在新版本部署前运行。在向上扩展期间，系统根本不处理配置文件。

此外，IIS 在容器命令运行前启动。此行为致使某些客户在容器命令中采取临时措施：在命令运行前暂停 IIS 服务器，命令完成后再次启动 IIS 服务器。

版本 1 修复了不一致问题，使 Windows Server 平台的行为与 Elastic Beanstalk 基于 Linux 的平台的行为更接近。在 v1 平台中，Elastic Beanstalk 始终在启动 IIS 服务器前运行容器命令。

v1 平台解决方案堆栈在 Windows Server 版本后带有 v1：

- 运行 IIS 8.5 的 64 位 Windows Server 2012 R2 v1.1.0
- 运行 IIS 8.5 的 64 位 Windows Server Core 2012 R2 v1.1.0
- 运行 IIS 8 的 64 位 Windows Server 2012 v1.1.0

- 运行 IIS 7.5 的 64 位 Windows Server 2008 R2 v1.1.0

此外，v1 平台将在运行容器命令前将应用程序源包的内容提取到 `C:\staging\`。容器命令完成后，此文件夹的内容会被压缩成 .zip 文件并部署到 IIS。此工作流程让您能够先使用命令或脚本修改应用程序源包的内容，然后再进行部署。

从 Windows Server 平台更早的主要版本迁移

在更新环境之前，先阅读此部分中的迁移注意事项。要将环境的平台更新为较新版本，请参阅更新 Elastic Beanstalk 环境的平台版本。

从 V1 到 V2

Windows Server 平台 v2 不支持 .NET 内核 1.x 和 2.0。如果您正在将应用程序从 Windows Server v1 迁移到 v2，并且您的应用程序使用其中一个 .NET 内核版本，则将您的应用程序更新到 v2 支持的 .NET 内核版本。有关受支持版本的列表，请参阅中的 AWS Elastic Beanstalk 平台中的 将 Windows Server 上的 .NET 与 IIS 结合使用。

如果您的应用程序使用自定义 Amazon Machine Image (AMI)，请基于 Windows Server 平台 v2 AMI 创建新的自定义 AMI。要了解更多信息，请参阅“使用自定义 Amazon 系统映像 (AMI)”。

> **Note**
>
> Windows Server v2 中新增的部署和更新功能依赖于增强型运行状况。当您将环境迁移到 v2 时，增强型运行状况处于禁用状态。启用它以使用这些功能。有关详细信息，请参阅启用 Elastic Beanstalk 增强型运行状况报告。

从 V1 之前的版本

除了从 v1 进行迁移的注意事项之外，如果要将应用程序从早于 v1 的 Windows Server 解决方案堆栈进行迁移，并且当前使用容器命令，请删除已添加的任何命令以解决迁移到更新版本时的处理不一致问题。从 v1 开始，容器命令确保在部署的应用程序源代码之前且在 IIS 启动之前运行完毕。这样，您就可以对 `C:\staging` 中的源代码进行任何更改，并在此步骤中修改 IIS 配置文件而不致出现问题。

例如，您可以使用从 Amazon S3 将 DLL 文件下载到您的应用程序源中： AWS CLI

`.ebextensions\copy-dll.config`

```
container_commands:
  copy-dll:
```

```
    command: aws s3 cp s3://DOC-EXAMPLE-BUCKET/dlls/large-dll.dll .\lib\
```

有关使用配置文件的更多信息，请参阅使用配置文件 (.ebextensions) 进行高级环境自定义。

使用部署清单运行多个应用程序和 ASP.NET 内核应用程序。

您可以使用部署清单告知 Elastic Beanstalk 如何部署您的应用程序。通过使用此方法，您无需使用 MSDeploy 为在网站根路径上运行的单个 ASP.NET 应用程序生成源包。相反，您可以使用清单文件在不同路径上运行多个应用程序。或者，您也可以告诉 Elastic Beanstalk 使用 ASP.NET Core 部署和运行应用程序。您也可以使用部署清单配置一个应用程序池，在其中运行您的应用程序。

部署清单向 Elastic Beanstalk 添加了对 .NET Core 应用程序的支持。您可以在不使用部署清单的情况下部署 .NET Framework 应用程序。但是，.NET Core 应用程序需要在 Elastic Beanstalk 上运行部署清单。使用部署清单时，请为每个应用程序创建一个站点存档，然后将该站点存档捆绑在包含部署清单的另一个 ZIP 存档中。

部署清单还增加了在不同路径上运行多个应用程序的能力。一个部署清单定义了一组部署目标，每个部署目标有一个站点存档和一个 IIS 应在其上运行部署清单的路径。例如，您可以在 /api 路径上运行 Web API，以服务异步请求，以及使用 API 的根路径上的 Web 应用程序。

您也可以使用部署清单通过在 IIS 或 Kestrel 中的应用程序池运行多个应用程序。您可以将应用程序池配置为定期重启应用程序、运行 32 位应用程序或使用特定版本的 .NET 框架运行时。

要进行完全自定义，您可以在 Windows 中编写自己的部署脚本， PowerShell 然后告诉 Elastic Beanstalk 要运行哪些脚本来安装、卸载和重启应用程序。

部署清单和相关功能需要 Windows Server 平台版本 1.2.0 或更新版本。

小节目录

.NET Core 应用程序

您可以使用部署清单在 Elastic Beanstalk 上运行 .NET Core 应用程序。.NET Core 是 .NET 的跨平台版本，它附带一个命令行工具 (dotnet)。您可以使用它生成一个应用程序、在本地运行该应用程序并做好发布该应用程序的准备。

> **Note**
>
> 请参阅教程：使用 Elastic Beanstalk 部署 ASP.NET Core 应用程序以获取在 Elastic Beanstalk 上使用部署清单运行 .NET Core 应用程序的教程和示例应用程序。

要在 Elastic Beanstalk 上运行 .NET Core 应用程序，您可以运行 `dotnet publish` 并将 ZIP 存档中的输出打包，而不包括任何包含的目录。将具有部署清单的源包中的站点存档与类型为 `aspNetCoreWeb` 的部署目标放在一起。

以下部署清单将在根路径上运行一个来自名为 `dotnet-core-app.zip` 的站点存档的 .NET 内核应用程序。

Example aws-windows-deployment-manifest.json-.NET 核心

```
{
  "manifestVersion": 1,
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "my-dotnet-core-app",
        "parameters": {
          "archive": "dotnet-core-app.zip",
          "iisPath": "/"
        }
      }
    ]
  }
}
```

将清单和站点存档捆绑在一个 ZIP 文档中，以创建源包。

Example dotnet-core-bundle.zip

```
.
|-- aws-windows-deployment-manifest.json
`-- dotnet-core-app.zip
```

该站点存档包含已编译的应用程序代码、依赖项和 `web.config` 文件。

Example dotnet-core-app.zip

```
.
|-- Microsoft.AspNetCore.Hosting.Abstractions.dll
|-- Microsoft.AspNetCore.Hosting.Server.Abstractions.dll
|-- Microsoft.AspNetCore.Hosting.dll
|-- Microsoft.AspNetCore.Http.Abstractions.dll
|-- Microsoft.AspNetCore.Http.Extensions.dll
|-- Microsoft.AspNetCore.Http.Features.dll
|-- Microsoft.AspNetCore.Http.dll
|-- Microsoft.AspNetCore.HttpOverrides.dll
|-- Microsoft.AspNetCore.Server.IISIntegration.dll
|-- Microsoft.AspNetCore.Server.Kestrel.dll
|-- Microsoft.AspNetCore.WebUtilities.dll
|-- Microsoft.Extensions.Configuration.Abstractions.dll
|-- Microsoft.Extensions.Configuration.EnvironmentVariables.dll
|-- Microsoft.Extensions.Configuration.dll
|-- Microsoft.Extensions.DependencyInjection.Abstractions.dll
|-- Microsoft.Extensions.DependencyInjection.dll
|-- Microsoft.Extensions.FileProviders.Abstractions.dll
|-- Microsoft.Extensions.FileProviders.Physical.dll
|-- Microsoft.Extensions.FileSystemGlobbing.dll
|-- Microsoft.Extensions.Logging.Abstractions.dll
|-- Microsoft.Extensions.Logging.dll
|-- Microsoft.Extensions.ObjectPool.dll
|-- Microsoft.Extensions.Options.dll
|-- Microsoft.Extensions.PlatformAbstractions.dll
|-- Microsoft.Extensions.Primitives.dll
|-- Microsoft.Net.Http.Headers.dll
|-- System.Diagnostics.Contracts.dll
|-- System.Net.WebSockets.dll
|-- System.Text.Encodings.Web.dll
|-- dotnet-core-app.deps.json
|-- dotnet-core-app.dll
|-- dotnet-core-app.pdb
|-- dotnet-core-app.runtimeconfig.json
`-- web.config
```

请参阅教程以获取完成示例。

运行多个应用程序

您可以定义多个部署目标，从而使用一个部署清单运行多个应用程序。

以下部署清单配置两个 .NET Core 应用程序。该`WebApiSampleApp`应用程序实现了一个简单的 Web API，并在`/api`路径上提供异步请求。`DotNetSampleApp` 应用程序是在根路径上服务请求的 Web 应用程序。

Example aws-windows-deployment-manifest.json-多个应用程序

```
{
  "manifestVersion": 1,
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "WebAPISample",
        "parameters": {
          "appBundle": "WebApiSampleApp.zip",
          "iisPath": "/api"
        }
      },
      {
        "name": "DotNetSample",
        "parameters": {
          "appBundle": "DotNetSampleApp.zip",
          "iisPath": "/"
        }
      }
    ]
  }
}
```

此处提供了一个具有多个应用场合的示例应用程序：

- 可部署的源代码包 --v2.zip dotnet-multiapp-sample-bundle
- 源代码-dotnet-multiapp-sample-source-v2.zip

配置应用程序池

您可以在 Windows 环境中支持多个应用程序。有两种方法可供选择：

- 您可以将 out-of-process 托管模式与 Kestrel 网络服务器配合使用。使用此模型，您可以配置多个应用程序以在一个应用程序池中运行。
- 您可以使用进程内托管模式。使用此模型，您可以使用多个应用程序池运行多个应用程序，每个池中只有一个应用程序。如果您使用的是 IIS 服务器并且需要运行多个应用程序，则必须使用此方法。

要将 Kesttrel 配置为在一个应用程序池中运行多个应用程序，请在 `hostingModel="OutofProcess"` 文件中添加 `web.config`。考虑以下示例。

Example web.config-适用于 Ke out-of-process strel 托管模型

```
<configuration>
<location path="." inheritInChildApplications="false">
<system.webServer>
<handlers>
<add
    name="aspNetCore"
    path="*" verb="*"
    modules="AspNetCoreModuleV2"
    resourceType="Unspecified" />
</handlers>
<aspNetCore
    processPath="dotnet"
    arguments=".\CoreWebApp-5-0.dll"
    stdoutLogEnabled="false"
    stdoutLogFile=".\logs\stdout"
    hostingModel="OutofProcess" />
</system.webServer>
</location>
</configuration>
```

Example aws-windows-deployment-manifest.json-多个应用程序

```
{
"manifestVersion": 1,
  "deployments": {"msDeploy": [
      {"name": "Web-app1",
        "parameters": {"archive": "site1.zip",
          "iisPath": "/"
        }
      },
      {"name": "Web-app2",
        "parameters": {"archive": "site2.zip",
          "iisPath": "/app2"
        }
      }
    ]
  }
}
```

IIS 不支持一个应用程序池中的多个应用程序，因为它使用进程内托管模型。因此，您需要通过将每个应用程序分配到一个应用程序池来配置多个应用程序。换句话说，只将一个应用程序分配到一个应用程序池。

您可以将 IIS 配置为在 aws-windows-deployment-manifest.json 文件中使用不同的应用程序池。在参考下一个示例文件时进行以下更新：

- 添加 iisConfig 部分，该部分包含称为 appPools 的子部分。
- 在 appPools 数据块中，列出应用程序池。
- 在 deployments 部分中，为每个应用程序定义 parameters 部分。
- 对于每个应用程序，parameters 部分都将指定一个存档、一个运行该存档的路径以及要在其中运行的 appPool 。

以下部署清单配置了两个应用程序池，它们每 10 分钟重新启动一次应用程序。他们还将应用程序附加到以指定路径运行的 .NET Framework Web 应用程序。

Example aws-windows-deployment-manifest.json-每个应用程序池一个应用程序

```
{
"manifestVersion": 1,
  "iisConfig": {"appPools": [
      {"name": "MyFirstPool",
       "recycling": {"regularTimeInterval": 10}
      },
      {"name": "MySecondPool",
       "recycling": {"regularTimeInterval": 10}
      }
     ]
    },
  "deployments": {"msDeploy": [
      {"name": "Web-app1",
        "parameters": {
           "archive": "site1.zip",
           "iisPath": "/",
           "appPool": "MyFirstPool"
           }
      },
      {"name": "Web-app2",
        "parameters": {
           "archive": "site2.zip",
```

```
            "iisPath": "/app2",
            "appPool": "MySecondPool"
          }
      }
     ]
    }
}
```

定义自定义部署

为了实现更多控制，您可以通过定义自定义部署 来完全自定义应用程序部署。

以下部署清单告知 Elastic Beanstalk 运行一个名为 `install` 的 `siteInstall.ps1` 脚本。此脚本在实例启动和部署期间安装网站。除此之外，部署清单还告诉 Elastic Beanstalk 在部署期间安装新版本之前uninstall运行脚本，restart并在管理控制台中选择 “重启应用服务器” 时重新启动应用程序的脚本。 AWS

Example aws-windows-deployment-manifest.json-自定义部署

```
{
  "manifestVersion": 1,
  "deployments": {
    "custom": [
      {
        "name": "Custom site",
        "scripts": {
          "install": {
            "file": "siteInstall.ps1"
          },
          "restart": {
            "file": "siteRestart.ps1"
          },
          "uninstall": {
            "file": "siteUninstall.ps1"
          }
        }
      }
    ]
  }
}
```

包括使用清单和脚本运行源包中的应用程序所需的任何项目。

Example C ustom-site-bundle .zip

```
.
|-- aws-windows-deployment-manifest.json
|-- siteInstall.ps1
|-- siteRestart.ps1
|-- siteUninstall.ps1
`-- site-contents.zip
```

向 .NET 应用程序环境中添加 Amazon RDS 数据库实例

您可以使用 Amazon Relational Database Service (Amazon RDS) 数据库实例来存储由应用程序收集和修改的数据。数据库可以耦合到您的环境并由 Elastic Beanstalk 进行管理，也可以被创建为解耦数据库并由另一项服务进行外部管理。本主题提供使用 Elastic Beanstalk 控制台创建 Amazon RDS 的说明。数据库将耦合到您的环境并由 Elastic Beanstalk 进行管理。有关将 Amazon RDS 与 Elastic Beanstalk 集成的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

小节目录

向环境中添加数据库实例

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

添加一个数据库实例大约需要 10 分钟。环境更新完成后，您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息：

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

有关与 Elastic Beanstalk 环境耦合的数据库实例配置的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

下载驱动程序

使用 `EntityFramework` 下载并安装 `NuGet` 程序包以及适合您开发环境的数据库驱动程序。

.NET 的常见实体框架数据库提供程序

- SQL Server – `Microsoft.EntityFrameworkCore.SqlServer`
- MySQL – `Pomelo.EntityFrameworkCore.MySql`

- PostgreSQL – `Npgsql.EntityFrameworkCore.PostgreSQL`

连接到数据库

Elastic Beanstalk 在环境属性中提供所连数据库实例的连接信息。使用 `ConfigurationManager.AppSettings` 可读取这些属性并配置数据库连接。

Example Helpers.cs - 连接字符串方法

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web;

namespace MVC5App.Models
{
  public class Helpers
  {
    public static string GetRDSConnectionString()
    {
      var appConfig = ConfigurationManager.AppSettings;

      string dbname = appConfig["RDS_DB_NAME"];

      if (string.IsNullOrEmpty(dbname)) return null;

      string username = appConfig["RDS_USERNAME"];
      string password = appConfig["RDS_PASSWORD"];
      string hostname = appConfig["RDS_HOSTNAME"];
      string port = appConfig["RDS_PORT"];

      return "Data Source=" + hostname + ";Initial Catalog=" + dbname + ";User ID=" +
 username + ";Password=" + password + ";";
    }
  }
}
```

使用连接字符串初始化您的数据库环境。

Example DBContext.cs

```
using System.Data.Entity;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace MVC5App.Models
{
  public class RDSContext : DbContext
  {
    public RDSContext()
      : base(GetRDSConnectionString())
    {
    }

    public static RDSContext Create()
    {
      return new RDSContext();
    }
  }
}
```

这些区域有： AWS Toolkit for Visual Studio

Visual Studio 提供了适用于不同编程语言和应用程序类型的模板。开始时，您可以使用其中任何一个模板。AWS Toolkit for Visual Studio 还提供了三个启动应用程序开发的项目模板：AWS Console Project、AWS Web Project 和 AWS Empty Project。在此示例中，您将创建一个新的 ASP.NET Web 应用程序。

创建一个新的 ASP.NET Web 应用程序项目

1. 在 Visual Studio 的 File（文件）菜单中，单击 New（新建），然后单击 Project（项目）。
2. 在 New Project（新建项目）对话框中，单击 Installed Templates（已安装的模板），单击 Visual C#，然后单击 Web。单击 ASP.NET Empty Web Application（ASP.NET 空 Web 应用程序），键入项目名，然后单击 OK（确定）。

运行项目

请执行下列操作之一：

1. 按 F5。
2. 从 Debug（调试）菜单选择 Start Debugging（开始调试）。

本地测试

Visual Studio 可让您轻松地在本地测试应用程序。要测试或者运行 ASP.NET Web 应用程序，您需要拥有 Web 服务器。Visual Studio 提供多个选项，如 Internet Information Services (IIS)、IIS Express 或者内置的 Visual Studio 开发服务器。要了解有关其中每个选项的详情以及确定最适合的选项，请参阅 Visual Studio for ASP.NET Web 项目中的 Web 服务器。

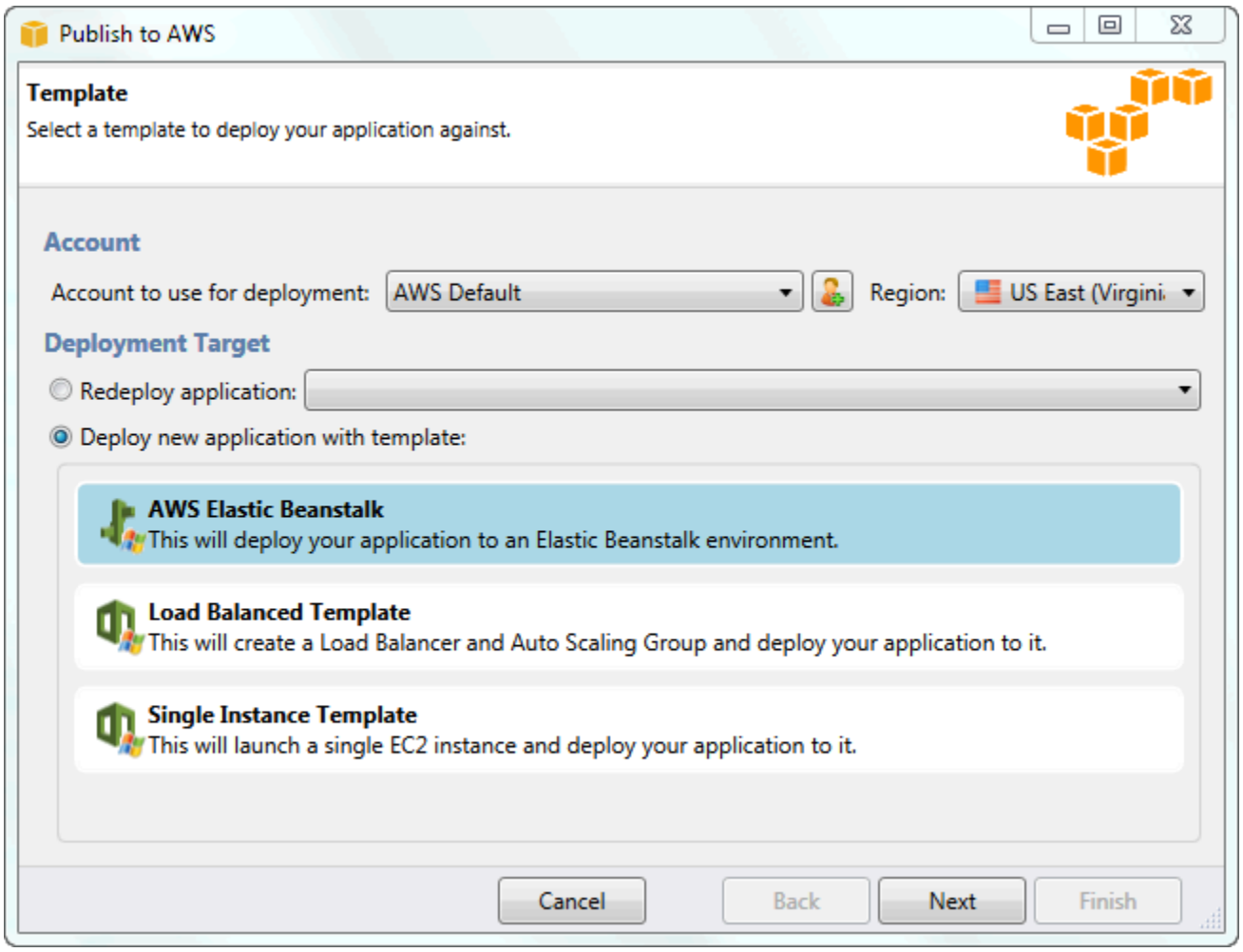
创建 Elastic Beanstalk 环境

在测试应用程序后，可随时将其部署到 Elastic Beanstalk。

> **Note**
>
> 配置文件需为存档中要包含的项目的一部分。或者，您也可以不将配置文件包含在项目中，而是使用 Visual Studio 将所有文件部署到项目文件夹中。在 Solution Explorer（解决方案资源管理器）中，右键单击项目名称，然后单击 Properties（属性）。单击 Package/Publish Web（程序包/发布 Web）选项卡。在 Items to deploy（要部署的项）部分中，选择下拉列表中的 All Files in the Project Folder（项目文件夹中的所有文件）。

使用 AWS Toolkit for Visual Studio 将您的应用程序部署到 Elastic Beanstalk

1. 在 Solution Explorer（解决方案资源管理器）中，右键单击您的应用程序，然后选择 Publish to AWS（发布到亚马逊云科技）。
2. 在 Publish to AWS (发布到亚马逊云科技) 向导中，输入您的账户信息。
   a. 对于 AWS account to use for deployment（要用于部署的亚马逊云科技账户），选择您的账户或选择 Other（其他）以输入新账户信息。
   b. 对于 Region（区域），选择要在其中部署应用程序的区域。有关可用 AWS 区域的信息，请参阅 AWS 一般参考 的 AWS Elastic Beanstalk 端点和配额。如果您选择了一个 Elastic Beanstalk 不支持的区域，则部署到 Elastic Beanstalk 的选项将变为不可用。
   c. 单击 Deploy new application with template（使用模板部署新应用程序）并选择 Elastic Beanstalk。然后单击 Next（下一步）。



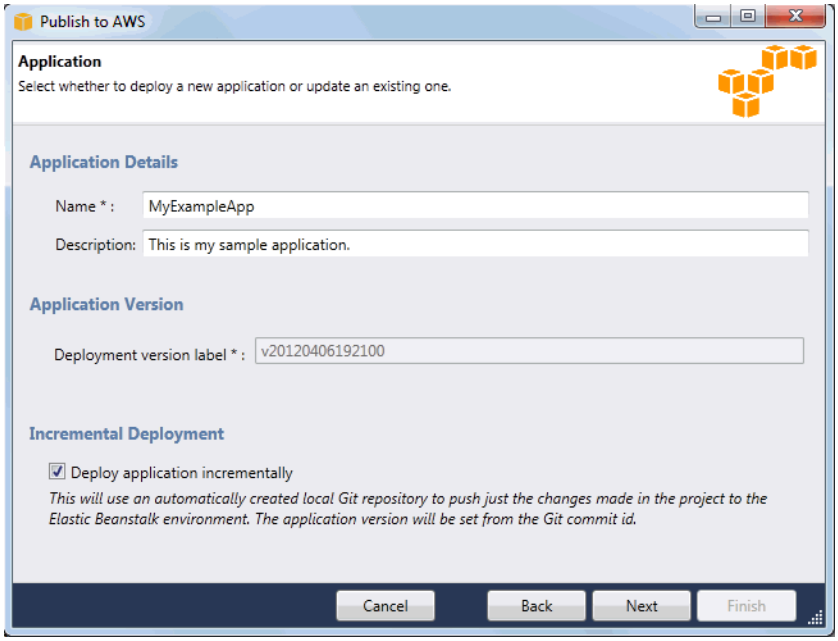
3. 在 Application（应用程序）页面上，输入应用程序详细信息。

   a. 对于 Name（名称），键入应用程序的名称。

   b. 对于 Description，键入应用程序的描述。此为可选步骤。

   c. 应用程序的版本标签将自动显示在 Deployment version label（部署版本标签）中。

   d. 选择 Deploy application incrementally（以递增方式部署应用程序），可仅部署已更改的文件。因为只需更新已更改的文件而不是所有文件，所以，增量部署的速度更快。如果您选择此选项，则系统会从 Git commit ID 设置应用程序版本。如果您选择不以增量的方式部署应用程序，那么，可以更新 Deployment version label（部署版本标签）框中的版本标签。



e. 单击 Next（下一步）。

4. 在 Environment（环境）页面上，描述环境详细信息。

a. 选择 Create a new environment for this application（为此应用程序创建新环境）。

b. 对于 Name（名称），键入环境的名称。

c. 对于 Description（描述），描述您的环境的特征。此为可选步骤。

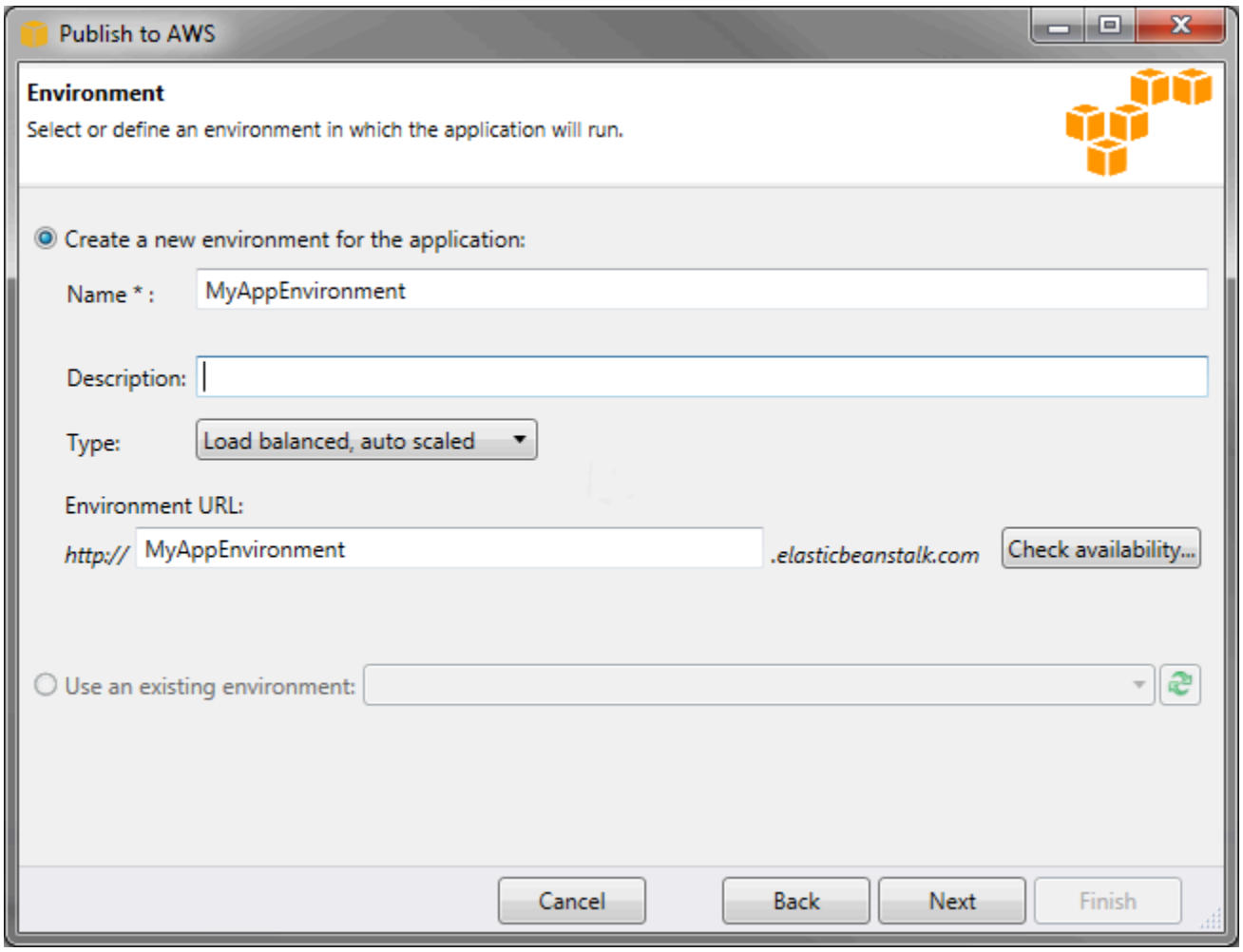
d. 选择需要的环境 Type（类型）。

可以选择 Load balanced（负载均衡）、auto scaled（自动扩展）或 Single instance（单一实例）环境。有关更多信息，请参阅环境类型。

> **Note**
>
> 对于单实例环境，负载均衡、自动扩展和运行状况检查 URL 设置不适用。

e. 将光标移到 Environment URL（环境 URL）框中后，环境 URL 会自动显示在该框中。

f. 单击 Check availability（检查可用性）按钮，确保环境 URL 处于可用状态。



g. 单击 Next (下一步)。

5. 在 AWS Options（亚马逊云科技选项）页面上，为部署配置额外的选项和安全信息。

   a. 对于 Container Type（容器类型），选择 64bit Windows Server 2012 running IIS 8（运行 IIS 8 的 64 位 Windows Server 2012）或 64bit Windows Server 2008 running IIS 7.5（运行 IIS 7.5 的 64 位 Windows Server 2008）。

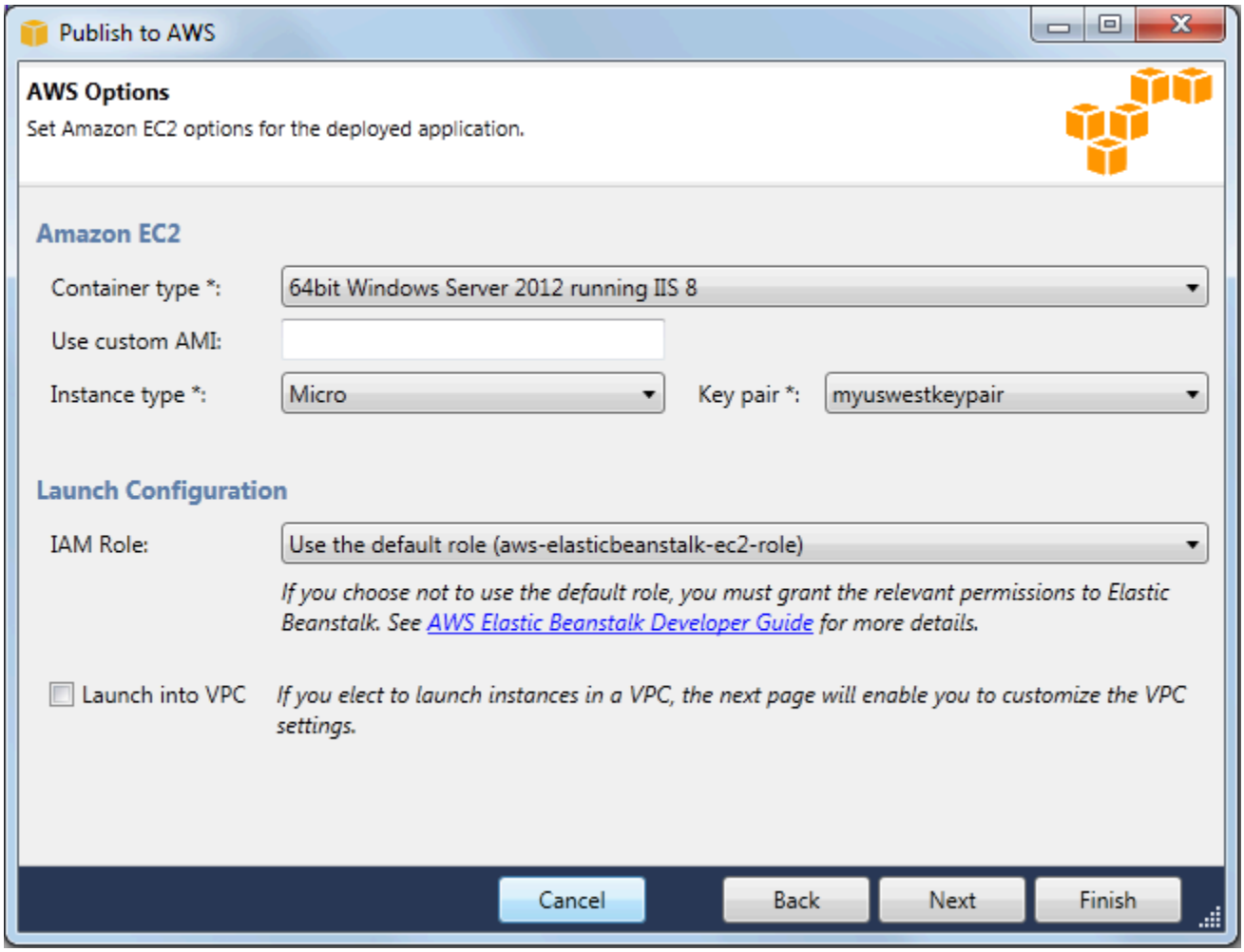
   b. 对于 Instance Type（实例类型），选择 Micro（微型）。

   c. 对于 Key pair (密钥对)，选择 Create new key pair (创建新密钥对)。键入新密钥对的名称（在本示例中，我们使用 **`myuswestkeypair`**），然后单击 OK（确定）。利用密钥对，可以对 Amazon EC2 实例进行远程桌面访问。有关 Amazon EC2 密钥对的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的使用凭证。

   d. 选择实例配置文件。

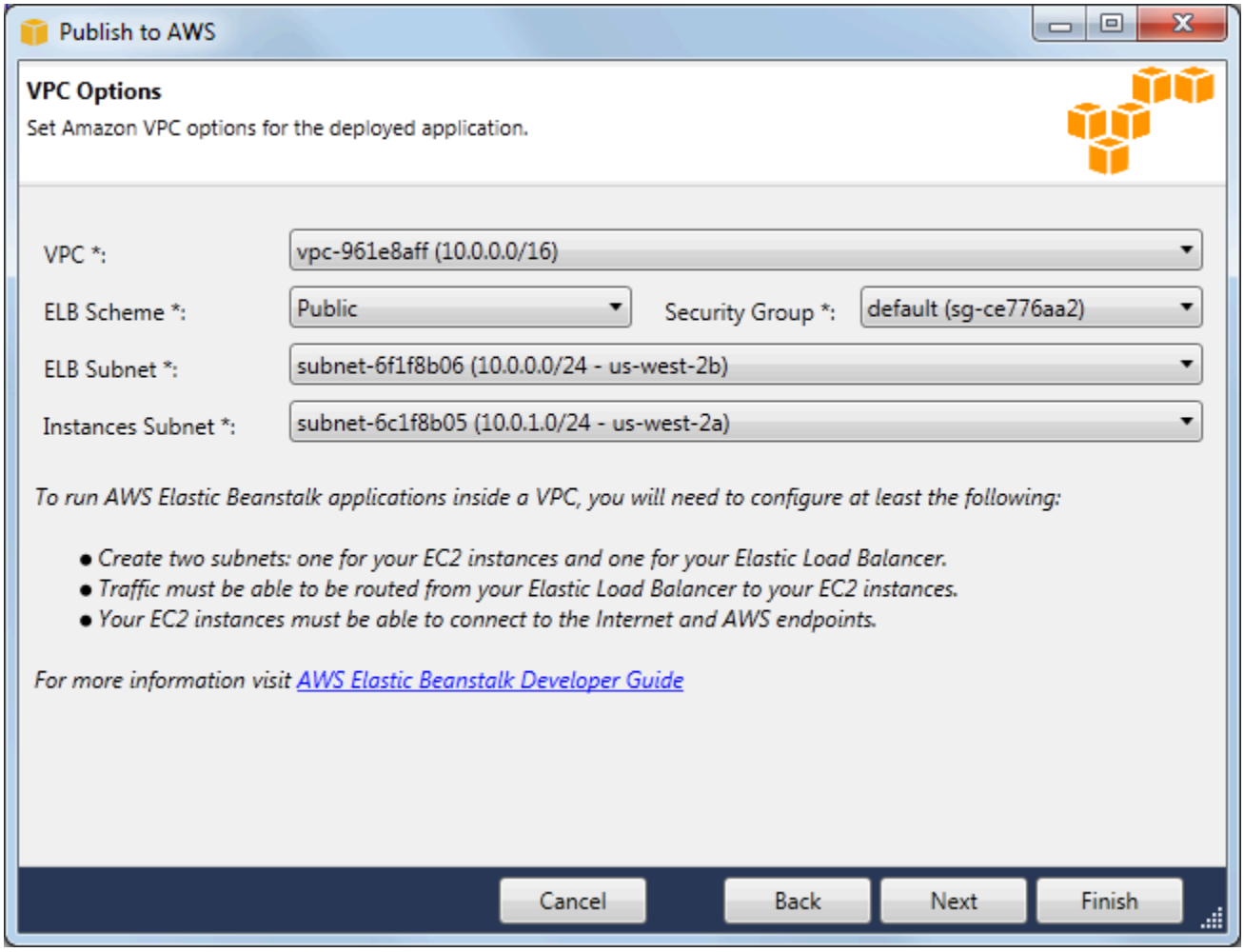
如果您没有实例配置文件，请选择 Create a default instance profile（创建默认实例配置文件）。有关将实例配置文件与 Elastic Beanstalk 结合使用的信息，请参阅管理 Elastic Beanstalk 实例配置文件。

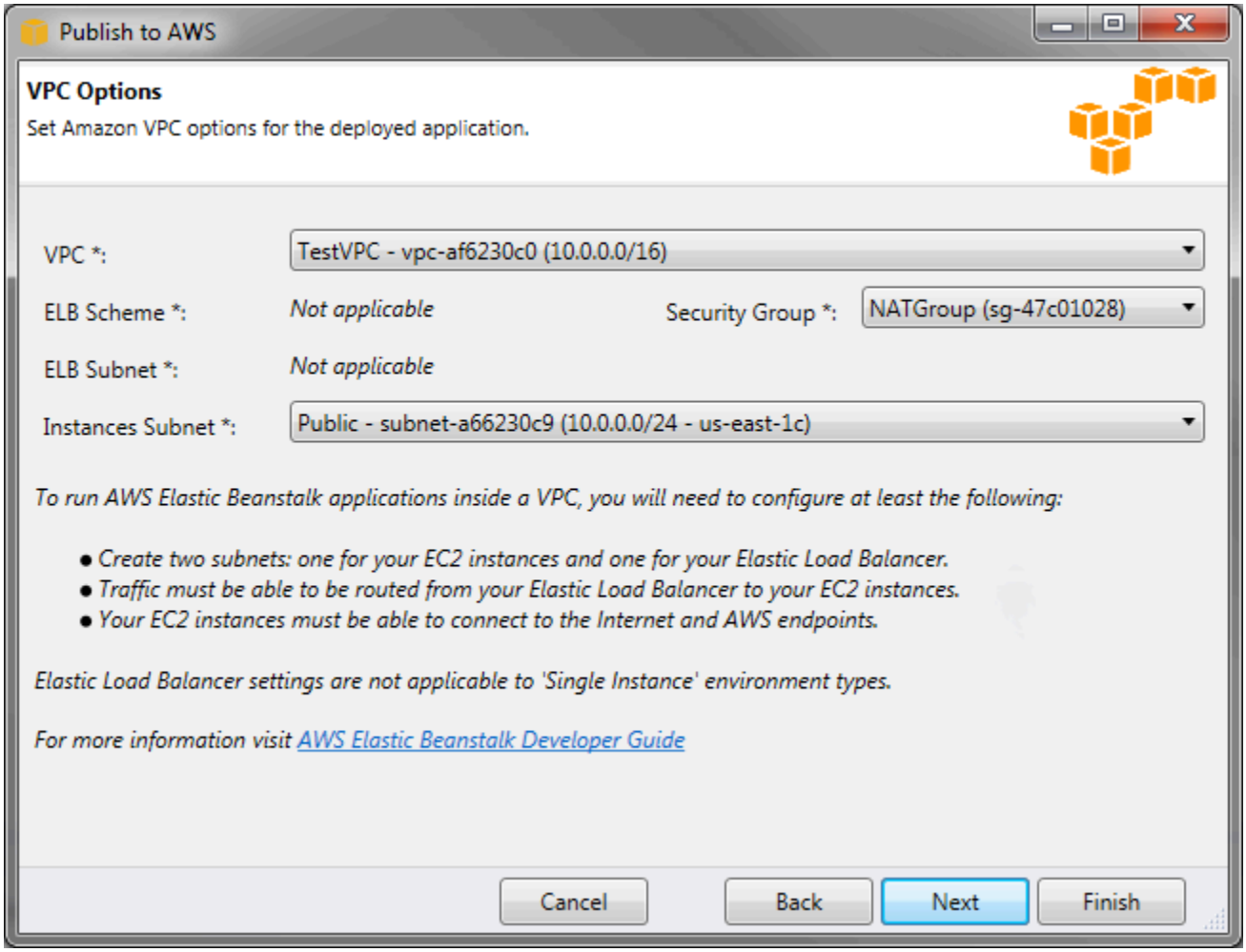
e. 如果您有要用于您环境中的自定义 VPC，请单击 Launch into VPC（在 VPC 中启动）。您可以在下一页上配置 VPC 信息。有关 Amazon VPC 的更多信息，请转到 Amazon Virtual Private Cloud (Amazon VPC)。有关支持的非旧式容器类型的列表，请参阅the section called "为什么某些平台版本标记为传统版本？"。



f. 单击 Next (下一步)。

6. 如果您选择在 VPC 内启动环境，则会显示 VPC Options（VPC 选项）页面；否则会显示 Additional Options（额外选项）页面。在这里可配置您的 VPC 选项。


Publish to AWS
VPC Options
Set Amazon VPC options for the deployed application.
VPC *:
vpc-961e8aff (10.0.0.0/16)
ELB Scheme *:
Public
Security Group *:
default (sg-ce776aa2)
ELB Subnet *:
subnet-6f1f8b06 (10.0.0.0/24 - us-west-2b)
Instances Subnet *:
subnet-6c1f8b05 (10.0.1.0/24 - us-west-2a)
To run AWS Elastic Beanstalk applications inside a VPC, you will need to configure at least the following:
Create two subnets: one for your EC2 instances and one for your Elastic Load Balancer.
Traffic must be able to be routed from your Elastic Load Balancer to your EC2 instances.
Your EC2 instances must be able to connect to the Internet and AWS endpoints.
For more information visit AWS Elastic Beanstalk Developer Guide
Cancel
Back
Next
Finish



a. 选择您要在其中启动环境的 VPC 的 VPC ID。

b. 对于负载均衡、可扩展的环境，如果您不希望弹性负载均衡器对 Internet 可用，请为 ELB 模式选择私有。

   对于单一实例环境，因为环境没有负载均衡器，所以此选项并不适用。有关更多信息，请参阅环境类型。

c. 对于负载均衡、可扩展的环境，为弹性负载均衡器和 EC2 实例选择子网。如果您创建了公有和私有子网，请确保弹性负载均衡器和 EC2 实例与正确的子网相关联。默认情况下，Amazon VPC 会使用 10.0.0.0/24 创建默认的公有子网，并使用 10.0.1.0/24 创建私有子网。您可以在 Amazon VPC 控制台 (https://console.aws.amazon.com/vpc/) 中查看现有子网。

   对于单一实例环境，您的 VPC 实例只需要一个公有子网供实例使用。因为环境没有负载均衡器，所以为负载均衡器选择子网并不适用。有关更多信息，请参阅环境类型。

d. 对于负载均衡、可扩展的环境，请选择您已为实例创建的安全组（如果适用）。

对于单一实例环境，您无需 NAT 设备。选择默认安全组。Elastic Beanstalk 为实例分配一个弹性 IP 地址，允许实例访问 Internet。

e. 单击 Next (下一步)。

7. 在 Application Options（应用程序选项）页上，配置您的应用程序选项。

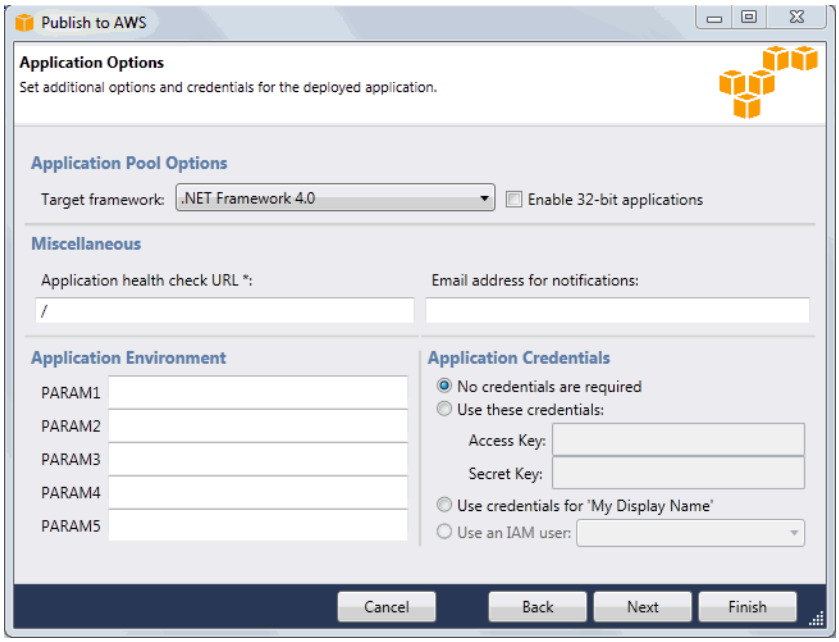
   a. 对于目标框架，选择 .NET Framework 4.0。

   b. Elastic Load Balancing 会通过运行状况检查确定运行您的应用程序的 Amazon EC2 实例是否正常。运行状况检查会按设置的时间间隔探测指定的 URL，从而确定该实例的状态。您可以通过在 Application health check URL（应用程序运行状况检查 URL）框中输入 URL（例如 `/myapp/index.aspx`）以覆盖默认 URL，使之对应于您的应用程序中的现有资源。有关应用程序运行状况检查的更多信息，请参阅“运行状况检查”。

   c. 如果要接收 Amazon Simple Notification Service (Amazon SNS) 通知，请键入电子邮件地址，这种通知会提醒您可能会影响应用程序的各种重大事件。

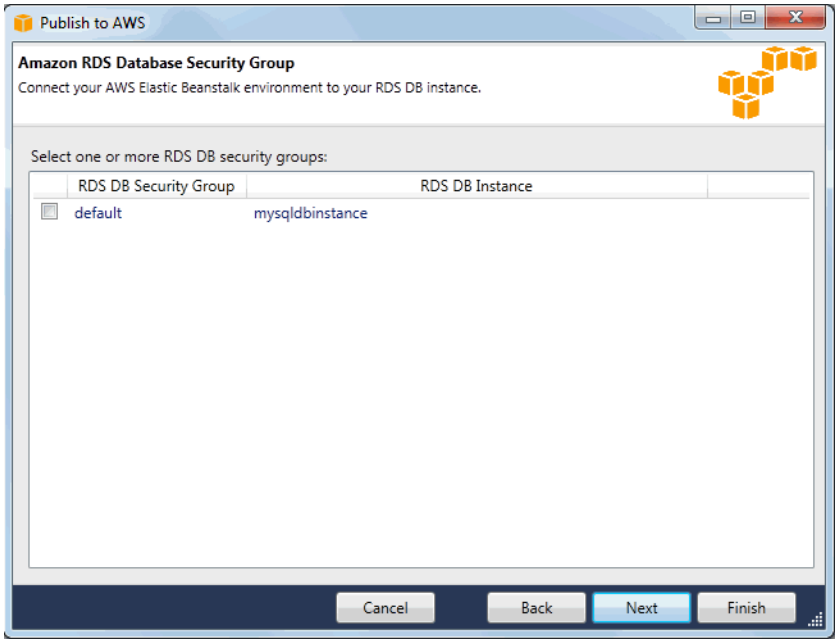
   d. Application Environment（应用程序环境）部分可让您在运行应用程序的 Amazon EC2 实例上指定环境变量。由于不再需要在环境中移动时重新编译资源代码，此设置可实现更高的可移植性。

   e. 选择您希望用于部署应用程序的应用程序凭证选项。

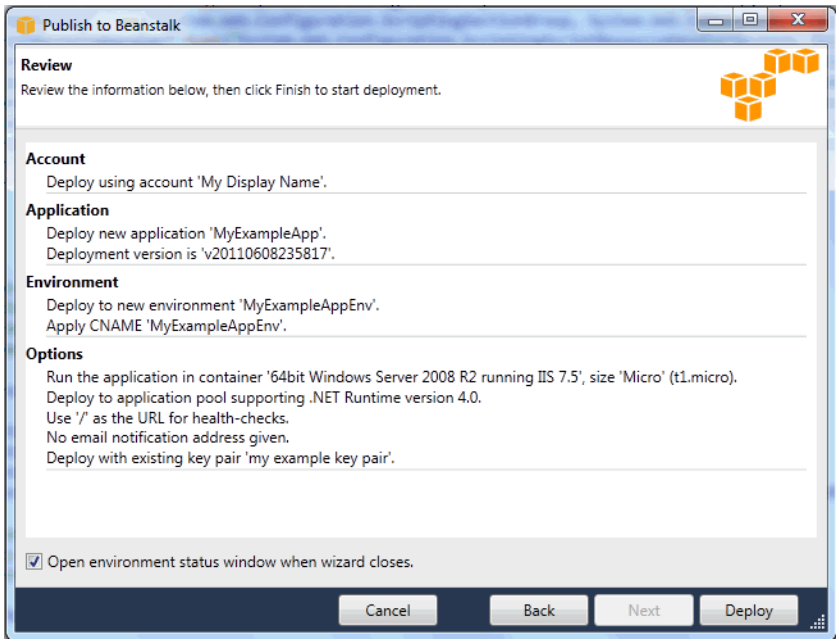


   f. 单击 Next (下一步)。

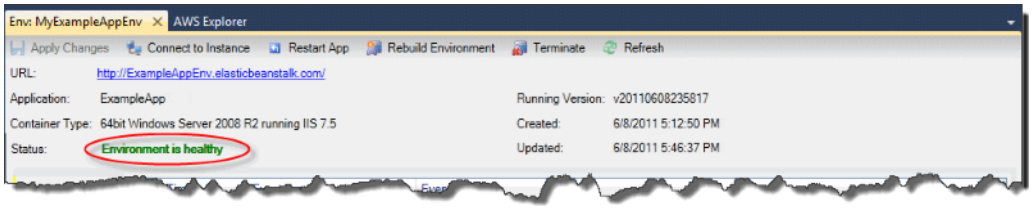
8. 如果之前设置了 Amazon RDS 数据库，则会显示 Amazon RDS DB Security Group（Amazon RDS 数据库安全组）页面。如果要将 Elastic Beanstalk 环境连接到 Amazon RDS 数据库实例，请选择一个或多个安全组。否则，请继续执行下一步。准备就绪时，单击 Next（下一步）。



9. 查看您的部署选项。如果所有内容都准确无误，请单击 Deploy (部署)。



您的 ASP.NET 项目会导出为 Web 部署文件，上传到 Amazon S3，并通过 Elastic Beanstalk 注册为新的应用程序版本。Elastic Beanstalk 部署功能将监控您的环境，直到部署了新代码的环境变得可用为止。在“env:<environment name>”（环境: <环境名称>）选项卡上，您将看到环境的状态。



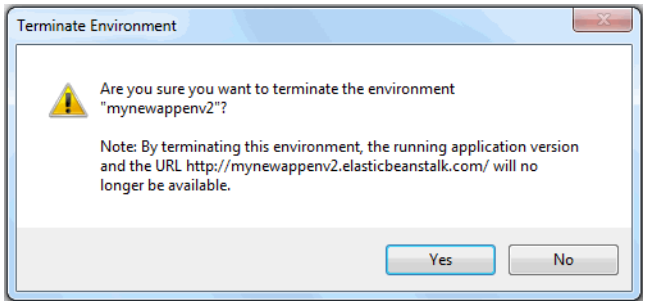
终止环境

为了避免因为未使用的 AWS 资源而产生费用，您可以使用 AWS Toolkit for Visual Studio 终止正在运行的环境。

> **Note**
>
> 稍后，您始终都可以使用相同的版本启动新的环境。

终止环境

1. 在 AWS Explorer 中，展开 Elastic Beanstalk 节点和应用程序节点。右键单击应用程序环境，选择 Terminate Environment（终止环境）。
2. 当系统提示时，单击 Yes（是）以确认要终止该环境。Elastic Beanstalk 需要几分钟时间才能终止环境中运行的 AWS 资源。



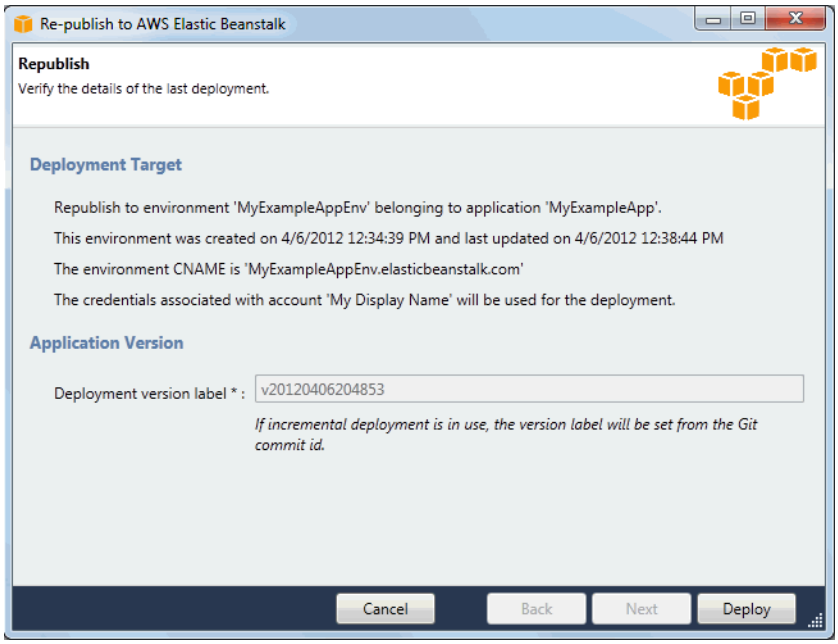
> **Note**
>
> 终止环境时，与已终止环境相关联的别名记录可供任何人使用。

部署到您的环境

在您的应用程序测试完毕后，就可以很轻松地编辑和重新部署您的应用程序，并在稍后查看相关结果。

编辑和重新部署您的 ASP.NET Web 应用程序

1. 在 Solution Explorer（解决方案资源管理器）中，右键单击您的应用程序，然后单击 Republish to Environment <***your environment name***>（重新发布到环境 <<replaceable>环境名称</replaceable>>）。此时，会打开 Re-publish to AWS Elastic Beanstalk（重新发布到 Amazon Elastic Beanstalk）向导。



2. 查看您的部署详细信息，然后单击 Deploy（部署）。

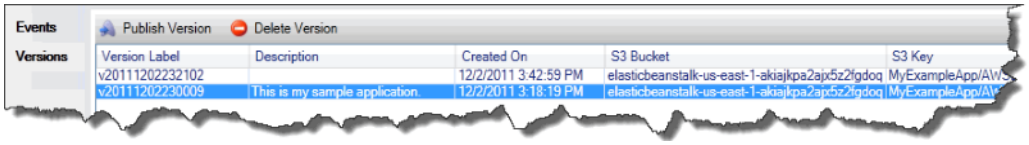
> **Note**
>
> 如果要更改任何设置，可以单击 Cancel（取消），并改为使用 Publish to AWS（发布到亚马逊云科技）向导。有关说明，请参阅创建 Elastic Beanstalk 环境。

您的更新 ASP.NET Web 项目会导出为 Web 部署文件 (其中带有新的版本标签)，上传到 Amazon S3，然后在 Elastic Beanstalk 中注册为新的应用程序版本。Elastic Beanstalk 部署功能会监控您的现有环境，直到该环境具有最新部署的代码并且变为可用状态为止。在 env:<***environment name***>（环境:<<replaceable>环境名称</replaceable>>）选项卡上，您将看到环境的状态。

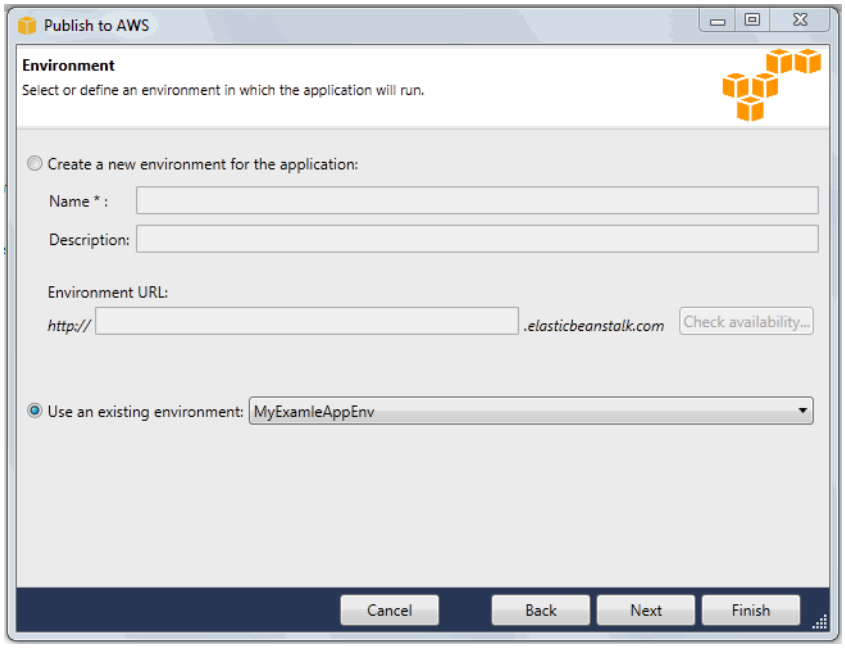
例如，如果您需要回滚到以前的应用程序版本，也可以将现有的应用程序部署到现有的环境中。

将应用程序版本部署到现有的环境中

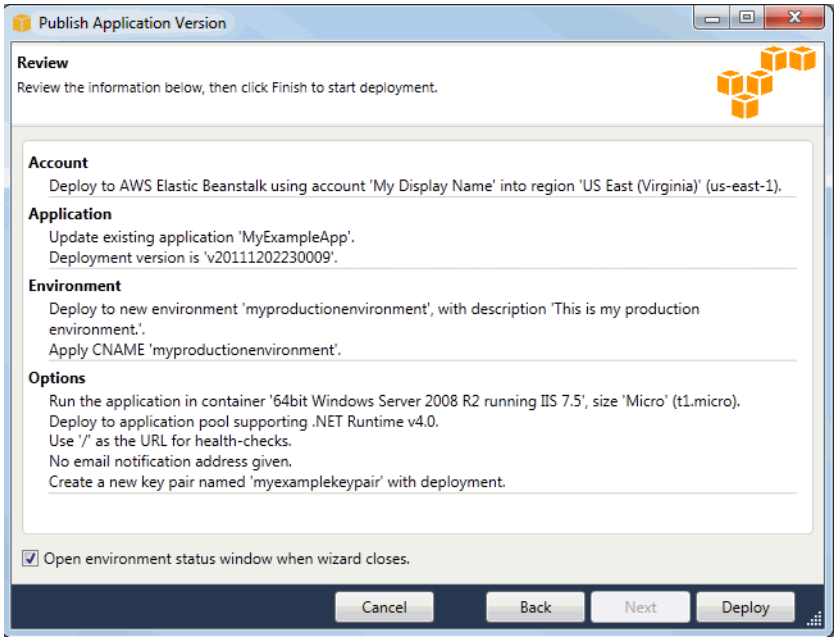
1. 通过在 AWS Explorer 中展开 Elastic Beanstalk 节点来右键单击您的 Elastic Beanstalk 应用程序。选择 View Status（查看状态）。
2. 在 App: <*application name*>（应用程序: <<replaceable>应用程序名称</replaceable>>）选项卡中，单击 Versions（版本）。



3. 单击您希望部署的应用程序版本，然后单击 Publish Version（发布版本）。
4. 在 Publish Application Version（发布应用程序版本）向导中，单击 Next（下一步）。



5. 查看您的部署选项，然后单击 Deploy（部署）。



您的 ASP.NET 项目将导出为 Web 部署文件，并上传到 Amazon S3。Elastic Beanstalk 部署功能将监控您的环境，直到部署了新代码的环境变得可用为止。在 env:<*environment name*>（环境:<<replaceable>环境名称</replaceable>>）选项卡上，您将看到环境的状态。

管理 Elastic Beanstalk 应用程序环境

通过 AWS Toolkit for Visual Studio 和 AWS 管理控制台，您可以更改应用程序环境使用的 AWS 资源的预置和配置。有关如何使用 AWS 管理控制台管理应用程序环境的详细信息，请参阅管理环境。本部分介绍可在 AWS Toolkit for Visual Studio 中编辑的特定服务设置，它也是应用程序环境配置的一部分。

更改环境配置设置

在部署应用程序时，Elastic Beanstalk 会配置许多 AWS 云计算服务。您可以使用 AWS Toolkit for Visual Studio 控制如何配置这些个别服务。

编辑应用程序的环境设置

- 展开 Elastic Beanstalk 节点和应用程序节点。然后在 AWS Explorer 中右键单击您的 Elastic Beanstalk 环境。选择 View Status (查看状态)。

  您现在可以配置以下各项的设置：

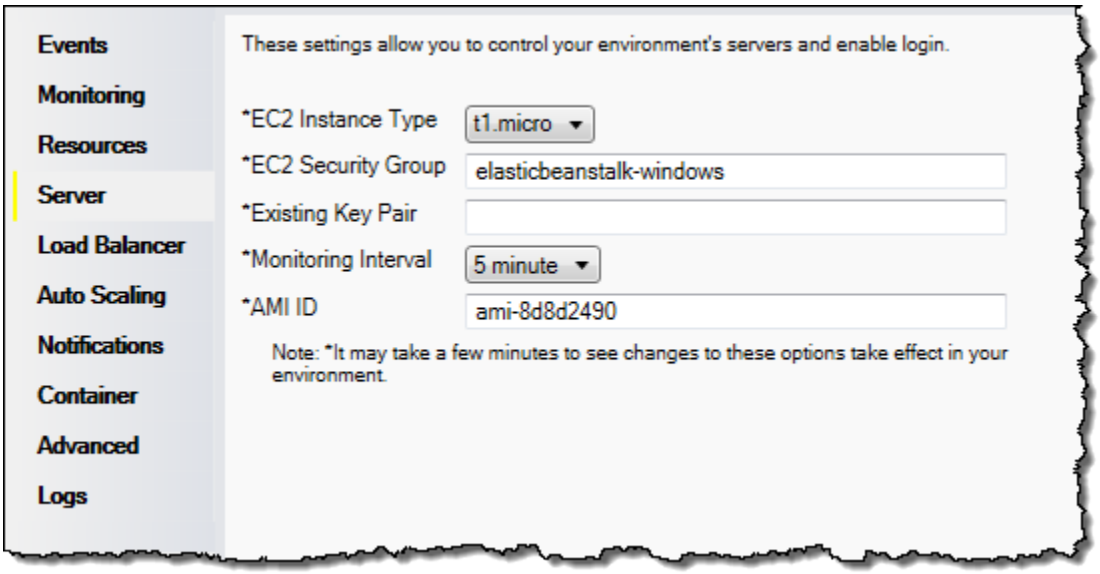
  - 服务器

- 负载均衡
- AutoScaling
- 通知
- 环境属性

使用 AWS Toolkit for Visual Studio 配置 EC2 服务器实例

Amazon Elastic Compute Cloud (Amazon EC2) 是一项 Web 服务，用于启动和管理 Amazon 数据中心里的服务器实例。您可以随时使用 Amazon EC2 服务器实例，并可根据需要在任意长的时间内使用，也可以用于任何合法用途。实例可以按照不同的规模和配置进行提供。有关更多信息，请转到 Amazon EC2。

您可以使用 AWS Toolkit for Visual Studio 的应用程序环境选项卡内的 Server（服务器）选项卡来编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。



Amazon EC2 实例类型

Instance type（实例类型）显示可用于您的 Elastic Beanstalk 应用程序的实例类型。请更改实例类型以选择特征（包括内存大小和 CPU 处理能力）最适合您的应用程序的服务器。例如，具有大量操作和长时间运行的操作的应用程序可能需要更多 CPU 或内存。

有关可用于 Elastic Beanstalk 应用程序的 Amazon EC2 实例类型的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的实例类型。

Amazon EC2 安全组

您可以使用 Amazon EC2 安全组 控制对 Elastic Beanstalk 应用程序的访问。安全组会定义实例的防火墙规则。这些规则会指定应将哪些进入 (即传入) 网络流量提交给实例。将丢弃所有其他进入流量。您可以随时针对不同的组修改这些规则。新的规则会自动在所有现在运行的和将来启动的实例上强制实施。

您可以使用 AWS 管理控制台或者 AWS Toolkit for Visual Studio 设置 Amazon EC2 安全组。通过在 EC2 Security Groups（EC2 安全组）文本框中输入一个或多个 Amazon EC2 安全组名称（用逗号分隔），您可以指定哪些 Amazon EC2 安全组控制对您的 Elastic Beanstalk 应用程序的访问。

> **Note**
>
> 如果要启用应用程序的运行状况检查，请确保可以从源 CIDR 范围 0.0.0.0/0 访问端口 80 (HTTP)。有关运行状况检查的详细信息，请参阅 运行状况检查。

使用 AWS Toolkit for Visual Studio 创建安全组

1. 在 Visual Studio 的 AWS Explorer 中，展开 Amazon EC2 节点，然后双击 Security Groups（安全组）。
2. 单击 Create Security Group（创建安全组），并输入安全组的名称和描述。
3. 单击 OK (确定)。

有关 Amazon EC2 安全组的更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的使用安全组。

Amazon EC2 密钥对

您可以使用 Amazon EC2 密钥对安全地登录到为 Elastic Beanstalk 应用程序预配置的 Amazon EC2 实例。

> **Important**
>
> 您必须创建 Amazon EC2 密钥对并将 Elastic Beanstalk 预配置的 Amazon EC2 实例配置为使用 Amazon EC2 密钥对，然后才能访问 Elastic Beanstalk 预配置的 Amazon EC2 实例。在向 Elastic Beanstalk 部署您的应用程序时，您可以使用 AWS Toolkit for Visual Studio 内的 Publish to AWS (发布到亚马逊云科技) 向导创建密钥对。如果要使用该 Toolkit 创建额外的密

钥对，请按照以下步骤操作。也可以使用 AWS 管理控制台设置 Amazon EC2 密钥对。有关为 Amazon EC2 创建密钥对的说明，请参阅 Amazon Elastic Compute Cloud 入门指南。

Existing Key Pair（现有密钥对）文本框可让您指定 Amazon EC2 密钥对的名称，您可以使用该密钥对安全地登录到运行 Elastic Beanstalk 应用程序的 Amazon EC2 实例。

指定 Amazon EC2 密钥对的名称

1. 展开 Amazon EC2 节点，双击 Key Pairs（密钥对）。
2. 单击 Create Key Pair（创建密钥对），输入密钥对名称。
3. 单击 OK（确定）。

有关 Amazon EC2 密钥对的更多信息，请转到 Amazon Elastic Compute Cloud 用户指南 中的使用 Amazon EC2 凭证。有关连接到 Amazon EC2 实例的更多信息，请参阅“列出和连接到服务器实例”。

监控间隔

默认情况下，仅启用基本 Amazon CloudWatch 指标。这些指标会以五分钟为周期返回数据。在 AWS Toolkit for Eclipse 中，您可为环境启用更精细的一分钟 CloudWatch 指标，方法为在环境的 Configuration（配置）选项卡的 Server（服务器）部分中为 Monitoring Interval（监控间隔）选择 1 minute（1 分钟）。

> **Note**
>
> 一分钟时间间隔指标可能产生 Amazon CloudWatch 服务费用。有关更多信息，请参阅 Amazon CloudWatch。

自定义 AMI ID

在 AWS Toolkit for Eclipse 中，您可将自定义 AMI 的标识符输入到环境的 Configuration（配置）选项卡的 Server（服务器）部分中的 Custom AMI ID（自定义 AMI ID）框，以便使用您自己的自定义 AMI 覆盖用于 Amazon EC2 实例的默认 AMI。

> **Important**
>
> 使用您自己的 AMI 是一项高级任务，应小心谨慎地执行。如果需要自定义 AMI，则建议您先使用默认 Elastic Beanstalk AMI，然后修改它。若要保持正常运行状态，Elastic Beanstalk 期望

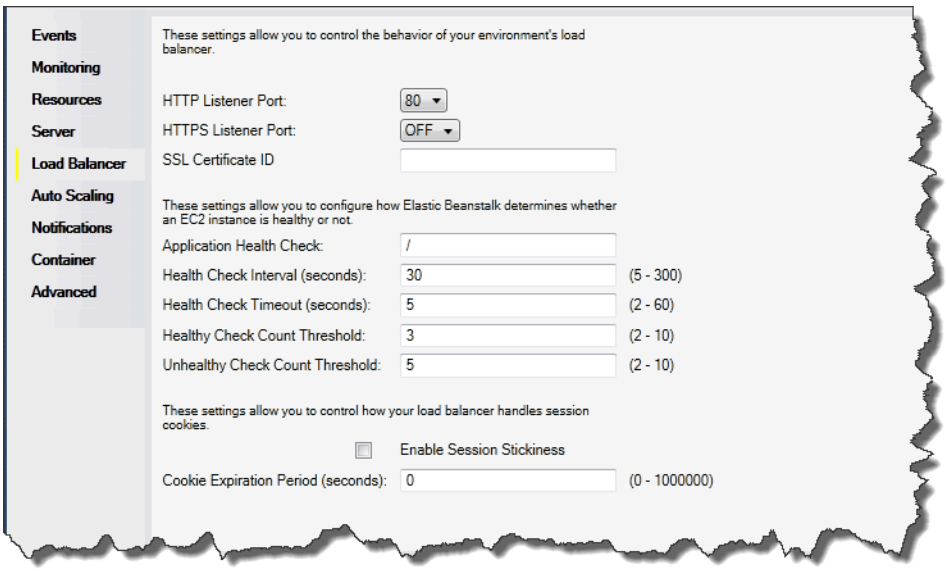
Amazon EC2 实例满足一系列要求，包括具有一个正在运行的主机管理器。如果未满足这些要求，您的环境可能无法正常运行。

使用 AWS Toolkit for Visual Studio 配置 Elastic Load Balancing

Elastic Load Balancing 是一种 Amazon Web 服务，可帮助您提高应用程序的可用性和可扩展性。该服务可让您轻松地在两个或更多的 Amazon EC2 实例之间分配应用程序负载。Elastic Load Balancing 通过冗余实现可用性，并支持应用程序的流量增长。

Elastic Load Balancing 可让您自动在运行的所有实例之间分配和平衡传入的应用程序流量。在您需要增加应用程序容量时，该服务还可让您轻松地添加新的实例。

Elastic Beanstalk 会在您部署应用程序时自动地预配置 Elastic Load Balancing。您可以使用 AWS Toolkit for Visual Studio 的应用程序环境选项卡内的 Load Balancer (负载均衡器) 选项卡来编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。



以下部分介绍了可为应用程序配置的 Elastic Load Balancing 参数。

端口

预配置来处理您的 Elastic Beanstalk 应用程序请求的负载均衡器会将请求发送到正在运行您的应用程序的 Amazon EC2 实例。预配置的负载均衡器会侦听 HTTP 和 HTTPS 端口上的请求，并将请求路由到 AWS Elastic Beanstalk 应用程序中的 Amazon EC2 实例。默认情况下，负载均衡器将处理 HTTP 端口上的请求。必须至少打开其中一个端口，要么是 HTTP 要么是 HTTPS。

> **Important**
>
> 确保您指定的端口没有锁定；否则，用户将无法连接到 Elastic Beanstalk 应用程序。

控制 HTTP 端口

若要关闭 HTTP 端口，请为 HTTP Listener Port（HTTP 侦听器端口）选择 OFF（关）。若要打开 HTTP 端口，需从列表中选择一个 HTTP 端口（例如，80）。

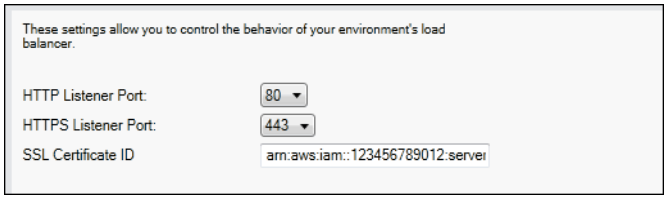
> **Note**
>
> 要使用默认端口 80 (如端口 8080) 以外的端口来访问您的环境，请将侦听器添加到现有负载均衡器并配置新侦听器来侦听该端口。
> 例如，当使用适用于 Classic Load Balancer 的 AWS CLI 时，键入以下命令可将 *LOAD_BALANCER_NAME* 替换为您用于 Elastic Beanstalk 的负载均衡器的名称。
>
> ```
> aws elb create-load-balancer-listeners --load-balancer-name LOAD_BALANCER_NAME
>  --listeners "Protocol=HTTP, LoadBalancerPort=8080, InstanceProtocol=HTTP,
>  InstancePort=80"
> ```
>
> 例如，当使用适用于 Application Load Balancer 的 AWS CLI 时，键入以下命令可将 *LOAD_BALANCER_ARN* 替换为您用于 Elastic Beanstalk 的负载均衡器的 ARN。
>
> ```
> aws elbv2 create-listener --load-balancer-arn LOAD_BALANCER_ARN --protocol HTTP
>  --port 8080
> ```
>
> 如果希望 Elastic Beanstalk 监控您的环境，请勿删除端口 80 上的侦听器。

控制 HTTPS 端口

Elastic Load Balancing 支持 HTTPS/TLS 协议，可为负载均衡器的客户端连接流量加密。从负载均衡器到 EC2 实例的连接使用明文加密。默认情况下，HTTPS 端口是关闭的。

打开 HTTPS 端口

1. 使用 AWS Certificate Manager (ACM) 创建新的证书，或者将证书和密钥上传到 AWS Identity and Access Management (IAM)。有关请求 ACM 证书的更多信息，请参阅 AWS Certificate Manager 用户指南中的请求证书。有关将第三方证书导入 ACM 中的更多信息，请参阅 AWS Certificate Manager 用户指南中的导入证书。如果 ACM 在您所在的区域不可用，请使用 AWS Identity and Access Management (IAM) 上传第三方证书。ACM 和 IAM 服务存储证书并为 SSL 证书提供 Amazon Resource Name (ARN)。有关创建证书并将证书上传到 IAM 的更多信息，请参阅 IAM 用户指南 中的使用服务器证书。
2. 通过为 HTTPS Listener Port（HTTP 侦听器端口）选择端口来指定 HTTPS 端口。



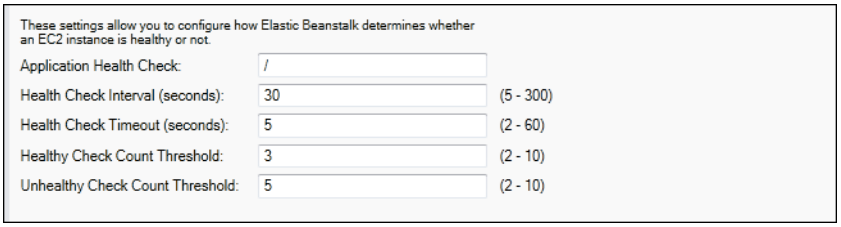
3. 对于 SSL Certificate ID（SSL 证书 ID），输入 SSL 证书的 Amazon Resource Name（ARN）。例如，**`arn:aws:iam::123456789012:server-certificate/abc/certs/build`** 或 **`arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`**。使用您在步骤 1 中创建或上传的 SSL 证书。

若要关闭 HTTPS 端口，请为 HTTPS Listener Port（HTTPS 侦听器端口）选择 OFF（关）。

运行状况检查

运行状况检查定义包括一个要用来查询实例运行状况的 URL。默认情况下，对于非早期容器，Elastic Beanstalk 使用 TCP:80，而对于早期容器，则使用 HTTP:80。您可以通过在 Application Health Check URL（应用程序运行状况检查 URL）框中输入 URL（例如 `/myapp/default.aspx`）来覆盖默认 URL，使之对应于您的应用程序中的现有资源。如果您覆盖默认 URL，则 Elastic Beanstalk 将使用 HTTP 来查询资源。要检查您使用的是否是早期容器类型，请参阅the section called “为什么某些平台版本标记为传统版本？”。

您可以使用 Load Balancing（负载均衡）面板的 EC2 Instance Health Check（EC2 实例运行状况检查）部分来控制运行状况检查的设置。



运行状况检查定义包括一个要用来查询实例运行状况的 URL。通过在 Application Health Check URL（应用程序运行状况检查 URL）框中输入 URL（例如 `/myapp/index.jsp`）来覆盖默认 URL，使之对应于您的应用程序中的现有资源。

下表介绍了可为您的应用程序设置的运行状况检查参数。

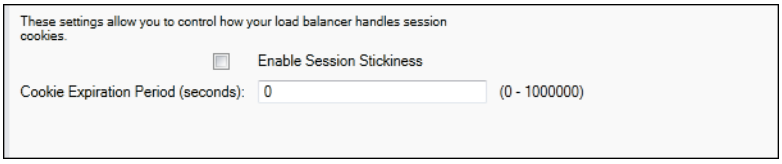
- 对于 Health Check Interval (seconds)（运行状况检查间隔(秒)），输入 Elastic Load Balancing 在对应用程序的 Amazon EC2 实例的运行状况进行检查之间等待的秒数。
- 对于 Health Check Timeout (seconds)（运行状况检查超时(秒)），指定 Elastic Load Balancing 在将实例视为无响应之前等待响应的秒数。
- 对于 Healthy Check Count Threshold (良好运行状况检查计数阈值) 和 Unhealthy Check Count Threshold (不佳运行状况检查计数阈值)，指定 Elastic Load Balancing 更改实例的运行状况状态之前连续的成功或失败 URL 探测的次数。例如，为 Unhealthy Check Count Threshold（不佳运行状况检查计数阈值）指定 **5**，即表示必须在该 URL 连续 5 次返回错误消息或超时后，Elastic Load Balancing 才将运行状况检查视为失败。

会话

默认情况下，负载均衡器会以最小的负载将每个请求独立地传送给该服务器实例。比较起来，粘性 (VPC) 会将用户的会话绑定到具体的服务器实例，以便该用户在会话期间发出的所有请求都会发送到同一个服务器实例中。

在为应用程序启用粘性会话后，Elastic Beanstalk 会使用负载均衡器生成的 HTTP Cookie。负载均衡器会使用负载均衡器生成的特别 Cookie 来跟踪每个请求的应用程序实例。在负载均衡器收到请求时，它首先会检查并查看请求中是否存在这个 Cookie。如果是这样的话，该请求会发送到 Cookie 中指定的应用程序实例。如果没有 Cookie，负载均衡器会根据现有的负载均衡算法选择一个应用程序实例。响应中会插入 Cookie，从而将同一用户发出的后续请求绑定到该应用程序实例中。策略配置会定义 Cookie 的到期时间，从而确定每个 Cookie 的有效持续时间。

您可以使用 Load Balancer（负载均衡器）选项卡上的 Sessions（会话）部分指定是否让应用程序的负载均衡器支持会话粘性。



有关 Elastic Load Balancing 的更多信息，请转到 Elastic Load Balancing 开发人员指南。

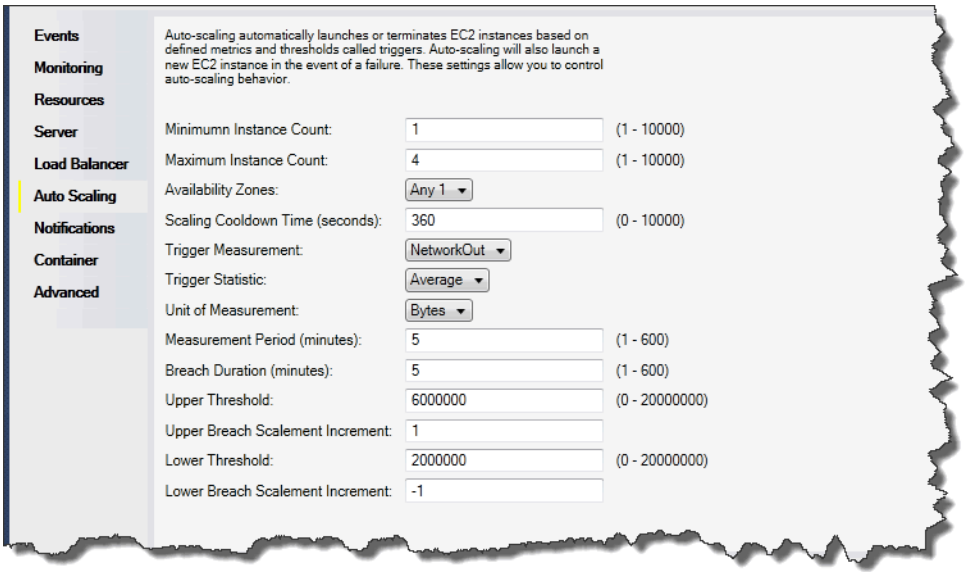
使用 AWS Toolkit for Visual Studio 配置 Auto Scaling

Amazon EC2 Auto Scaling 是一项 Amazon Web 服务，旨在根据用户定义的触发器自动启动或终止 Amazon EC2 实例。用户可以设置 Auto Scaling 组 并将 触发器 与这些组关联，以根据带宽使用量或 CPU 利用率等指标自动扩展计算资源。Amazon EC2 Auto Scaling 与 Amazon CloudWatch 协作，检索运行应用程序的服务器实例的指标。

借助 Amazon EC2 Auto Scaling，您可以获取一组 Amazon EC2 实例并设置各种参数，使此组的数量自动增加或减少。Amazon EC2 Auto Scaling 可以在该组中添加或删除 Amazon EC2 实例，以帮助您无缝处理应用程序的流量变化。

Amazon EC2 Auto Scaling 还会监控其启动的每个 Amazon EC2 实例的运行状况。如果任何实例意外终止，Amazon EC2 Auto Scaling 会检测到终止情况并启动替换实例。这一功能可让您自动维护固定的、预期数量的 Amazon EC2 实例。

Elastic Beanstalk 为您的应用程序预配置 Amazon EC2 Auto Scaling。您可以使用 AWS Toolkit for Visual Studio 的应用程序环境选项卡内的 Auto Scaling 选项卡来编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

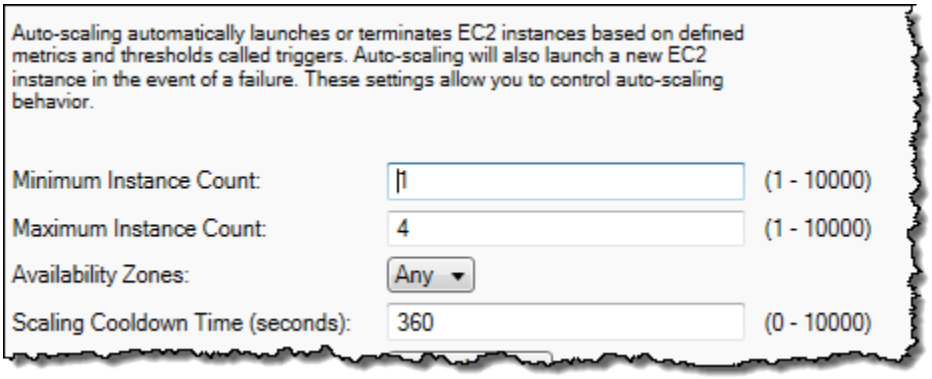


以下部分介绍了如何配置您的应用程序的 Auto Scaling 参数。

启动配置

您可以编辑启动配置以控制 Elastic Beanstalk 应用程序如何预配置 Amazon EC2 Auto Scaling 资源。

Minimum Instance Count（最小实例计数）和 Maximum Instance Count（最大实例计数）框可让您指定 Elastic Beanstalk 应用程序使用的 Auto Scaling 组的最小大小和最大大小。



> **Note**
>
> 要保持固定数量的 Amazon EC2 实例，请将 Minimum Instance Count（最小实例计数）和 Maximum Instance Count（最大实例计数）设置为相同的值。

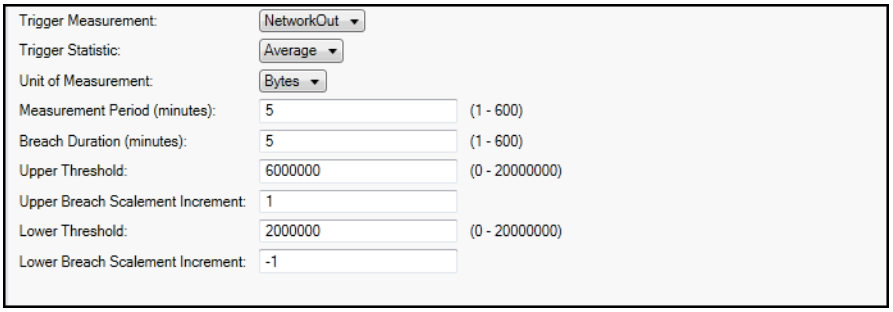
Availability Zones（可用区）框可让您指定希望 Amazon EC2 实例所在的可用区数。如果要构建容错的应用程序，则设置这个数字是十分重要的。如果一个可用区域出现故障，您的实例仍然会在其他可用区域上运行。

> **Note**
>
> 目前，您无法指定您的实例将放入哪些可用区域。

触发

触发器 是一种 Amazon EC2 Auto Scaling 机制，您可以通过设置该机制告知系统，什么时候希望增加（扩展）实例的数量，以及什么时候希望减少（缩减）实例的数量。您可以配置这些触发器，使其在将任何指标（如 CPU 利用率）发布到 Amazon CloudWatch 时激发，并确定是否已满足您指定的条件。当在指定的时间期限内超过为该指标指定的条件上限或者下限时，该触发会启动名为扩展活动 的长期运行流程。

您可以使用 AWS Toolkit for Visual Studio 为 Elastic Beanstalk 应用程序定义扩展触发器。



Amazon EC2 Auto Scaling 会根据实例的具体 Amazon CloudWatch 指标而触发各种操作。触发包括 CPU 使用率、网络流量和磁盘活动。使用 Trigger Measurement (触发测量标准) 设置选择触发的指标。

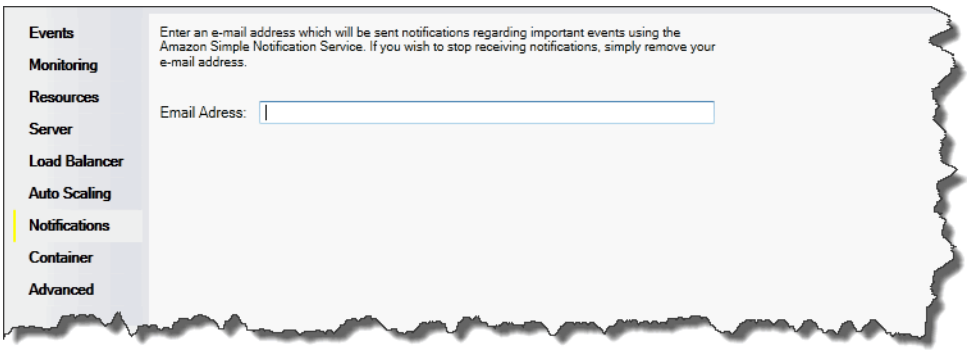
下表介绍了您可以使用 AWS 管理控制台配置的触发参数。

- 您可以指定该触发应该使用的统计数据。可以为 Trigger Statistic（触发统计数据）选择 Minimum（最小值）、Maximum（最大值）、Sum（总计）或 Average（平均值）。
- 对于 Unit of Measurement（测量单位），指定触发测量单位。
- 测量周期框内的值指定了 Amazon CloudWatch 对触发指标进行测量的频率。Breach Duration（违例持续时间）是激活触发器之前，指标可以超出所定义限制范围 [通过 Upper Threshold（上限）和 Lower Threshold（下限）指定] 的时长。
- 对于 Upper Breach Scale Increment (上限违例扩展增量) 和 Lower Breach Scale Increment (下限违例扩展增量)，指定执行扩展活动时要添加或删除的 Amazon EC2 实例数。

有关 Amazon EC2 Auto Scaling 的更多信息，请参阅 Amazon Elastic Compute Cloud 文档上的 Amazon EC2 Auto Scaling 部分。

使用 AWS Toolkit for Visual Studio 配置通知

Elastic Beanstalk 使用 Amazon Simple Notification Service (Amazon SNS) 向您通知影响应用程序的重要事件。要启用 Amazon SNS 通知，只需在 Email Address（电子邮件地址）框中输入您的电子邮件地址。若要禁用这些通知，请从框中删除您的电子邮件地址。



使用 AWS Toolkit for Visual Studio 配置 .NET 容器

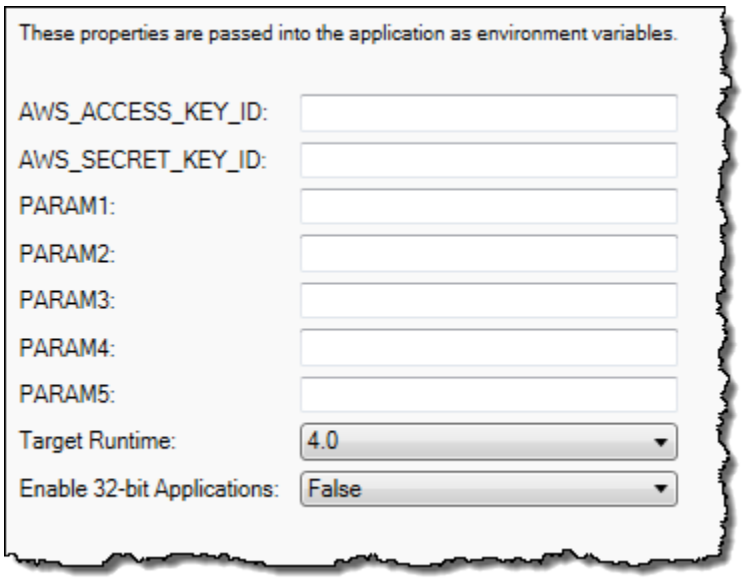
Container/.NET Options（容器/.NET 选项）面板可让您调整 Amazon EC2 实例的性能，以及启用或禁用 Amazon S3 日志交替。您可以使用 AWS Toolkit for Visual Studio 配置您的容器信息。

> **Note**
>
> 您可以通过切换环境别名记录的方法将配置设置修改为零故障时间。有关更多信息，请参阅使用 Elastic Beanstalk 进行蓝/绿部署。

如果需要，则可以扩展参数数量。有关扩展参数的信息，请参阅选项设置。

访问 Elastic Beanstalk 应用程序的“容器/.NET 选项”面板

1. 在 AWS Toolkit for Visual Studio 中，展开 Elastic Beanstalk 节点和应用程序节点。
2. 在 AWS Explorer 中，双击您的 Elastic Beanstalk 环境。
3. 在 Overview（概述）窗格底部，单击 Configuration（配置）选项卡。
4. 在 Container（容器）下，您可以配置容器的选项。

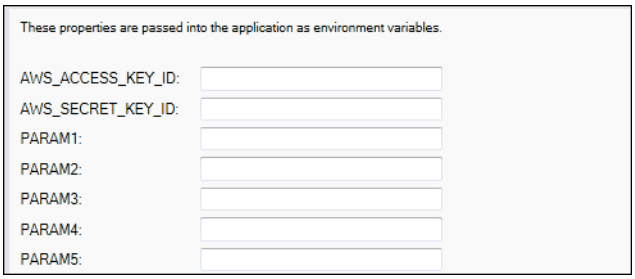


.NET 容器选项

您可以选择应用程序的 .NET Framework 版本。为 Target runtime（目标运行时）选择 2.0 或 4.0。如果要启用 32 位应用程序，请选择 Enable 32–bit Applications（启用 32 位应用程序）。

应用程序设置

Application Settings（应用程序设置）部分可让您指定您可从应用程序代码中读取的环境变量。



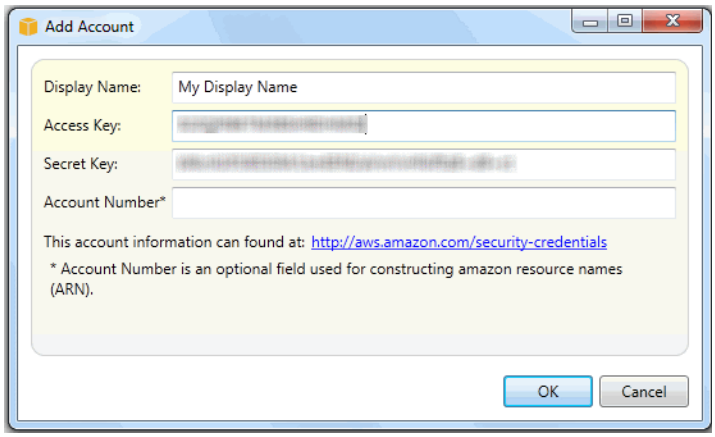
管理账户

如果要设置不同的 AWS 账户以便执行不同的任务，如测试、暂存和生产，您可以使用 AWS Toolkit for Visual Studio 添加、编辑和删除账户。

管理多个账户

1. 在 Visual Studio 的 View（视图）菜单中，单击 AWS Explorer。
2. 在账户列表旁边，单击添加账户按钮。

此时显示添加账户对话框。



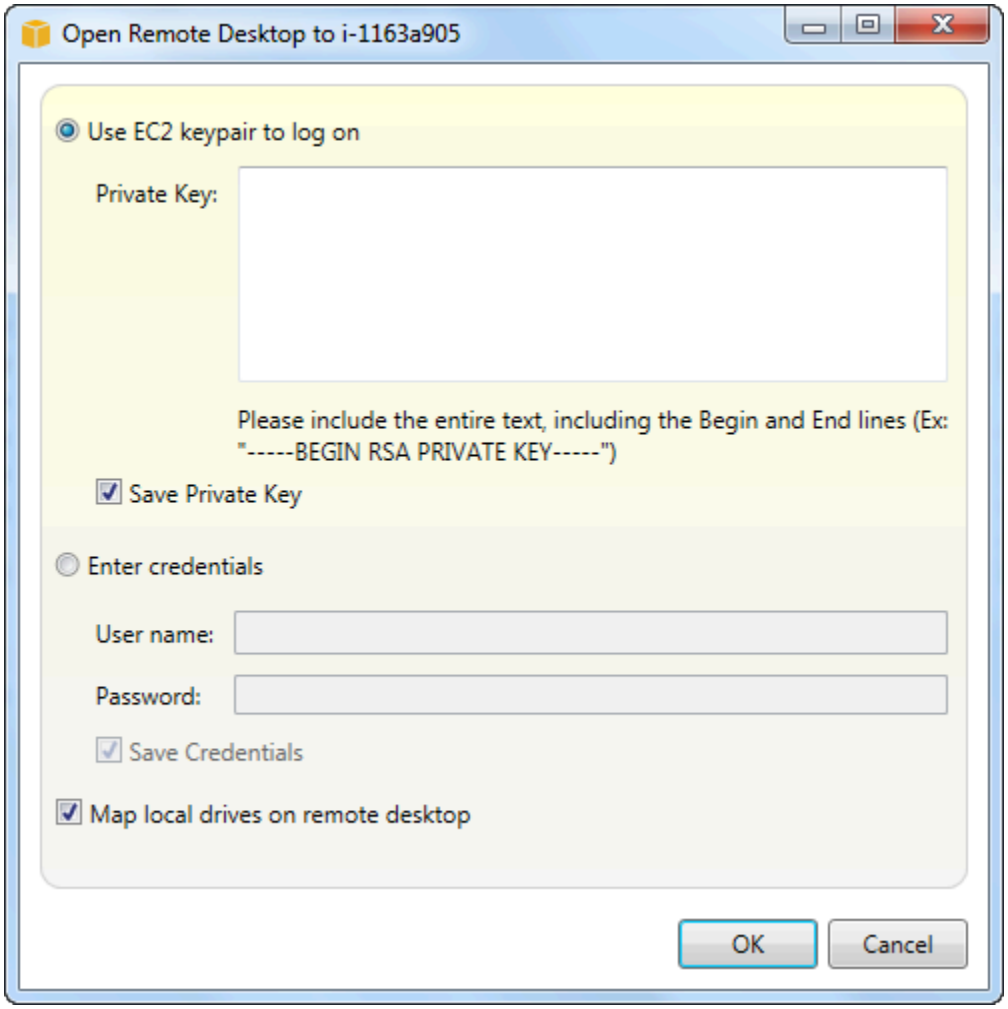
3. 填写所需的信息。
4. 现在，您的账户信息会显示在 AWS Explorer 选项卡中。在向 Elastic Beanstalk 发布时，您可以选择希望使用的账户。

列出和连接到服务器实例

您可以通过 AWS Toolkit for Visual Studio 或者从 AWS 管理控制台查看一系列运行 Elastic Beanstalk 应用程序环境的 Amazon EC2 实例。您可以使用远程桌面连接到这些实例。有关使用 AWS 管理控制台列出和连接到服务器实例的详细信息，请参阅列出和连接到服务器实例。以下部分逐步介绍了如何使用 AWS Toolkit for Visual Studio 查看和连接服务器实例。

查看和连接到环境的 Amazon EC2 实例

1. 在 Visual Studio 的 AWS Explorer 中，展开 Amazon EC2 节点，然后双击 Instances（实例）。
2. 在 Instance（实例）栏中，右键单击在应用程序的负载均衡器中运行的 Amazon EC2 实例的 ID，并在上下文菜单中选择 Open Remote Desktop（打开远程桌面）。



3. 选择 Use EC2 keypair to log on（使用 EC2 密钥对进行登录），并将用于部署应用程序的私有密钥文件内容粘贴到 Private key（私有密钥）框中。另一种方法是，在 User name（用户名）和 Password（密码）文本框中，输入您的用户名和密码。

> **Note**
>
> 如果密钥对保存在工具包内，那么不会显示该文本框。

4. 单击 OK (确定)。

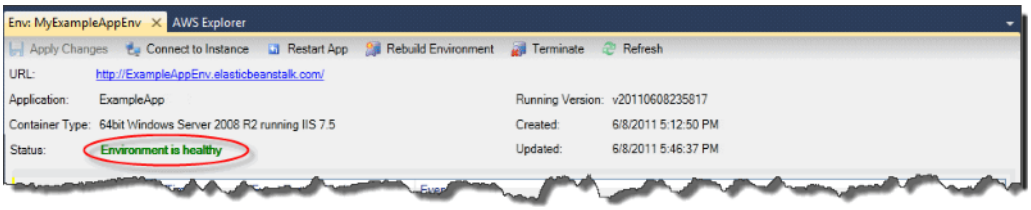
监控应用程序运行状况

当您运行生产网站时，了解您的应用程序是否可用以及能否对请求做出响应非常重要。为了帮助监控应用程序的响应能力，Elastic Beanstalk 提供了一些功能，您可以在其中监控有关应用程序的统计数据并创建在超过阈值时触发的警报。

有关 Elastic Beanstalk 提供的运行状况监控的信息，请参阅基本运行状况报告。

您可以使用 AWS Toolkit for Visual Studio 或者 AWS 管理控制台访问关于应用程序的操作信息。

该工具包会在状态字段中显示您环境的状态和应用程序运行状况。



监控应用程序运行状况

1. 在 AWS Toolkit for Visual Studio 的 AWS Explorer 中，展开 Elastic Beanstalk 节点，然后展开您的应用程序节点。
2. 右键单击您的 Elastic Beanstalk 环境，然后单击 View Status（查看状态）。
3. 在应用程序环境选项卡上，单击监控。

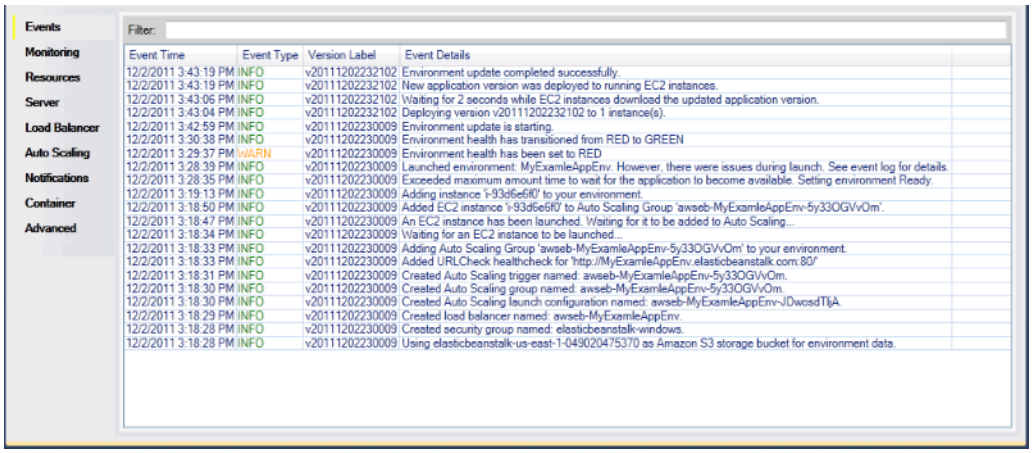
   监控面板包含一组图表，用于显示您特定应用程序环境的资源使用情况。

> **Note**
>
> 默认情况下，时间范围设置为最近一个小时。要修改这个设置，请在时间范围列表中，单击另一个时间范围。

您可以使用 AWS Toolkit for Visual Studio 或者 AWS 管理控制台，查看与您的应用程序相关联的事件。

查看应用程序事件

1. 在 AWS Toolkit for Visual Studio 的 AWS Explorer 中，展开 Elastic Beanstalk 节点和您的应用程序节点。
2. 在 AWS Explorer 中，右键单击 Elastic Beanstalk 环境，然后单击 View Status（查看状态）。
3. 在应用程序环境选项卡上，单击事件。



使用部署工具在 .NET 中部署 Elastic Beanstalk 应用程序

AWS Toolkit for Visual Studio 包含一个部署工具，这是与 AWS Toolkit 中的部署向导具有相同功能的命令行工具。您可在构建管道或其他脚本中使用部署工具，以自动实施到 Elastic Beanstalk 的部署。

该部署工具支持初始部署和重新部署。如果您原先已使用该部署工具部署应用程序，那么您可以使用 Visual Studio 中的部署向导进行重新部署。类似地，如果您已经使用了向导进行部署，那么您可以使用部署工具进行重新部署。

> **Note**
>
> 部署工具不会像控制台或 EB CLI 那样为配置选项应用建议值。使用配置文件可以确保在您启动环境时配置了所需的全部设置。

本章将引导您使用该部署工具将一个示例 .NET 应用程序部署到 Elastic Beanstalk，然后运用增量式部署进行重新部署。有关该部署工具的更为深入的讨论（包括参数选项），请参阅部署工具。

先决条件

要使用该部署工具，您需要安装 AWS Toolkit for Visual Studio。有关先决条件和安装说明的信息，请参阅 AWS Toolkit for Microsoft Visual Studio。

该部署工具一般安装在 Windows 的以下一个目录下：

32 位	64 位
C:\Program Files\AWS Tools\Deployment Tool\awsdeploy.exe	C:\Program Files (x86)\AWS Tools\Deployment Tool\awsdeploy.exe

部署到 Elastic Beanstalk

要使用该部署工具将示例应用程序部署到 Elastic Beanstalk，您首先需要修改 ElasticBeanstalkDeploymentSample.txt 配置文件，该文件位于 Samples 目录下。该配置文件包含部署应用程序所需的必要信息，包括应用程序名称、应用程序版本、环境名称以及您的 AWS 访问凭证。修改配置文件后，使用命令行部署示例应用程序。您的 Web 部署文件会上传到 Amazon S3，并通过 Elastic Beanstalk 注册为一个新的应用程序版本。部署应用程序需要花几分钟时间。只要环境健康，该部署工具就会输出正在运行的应用程序的 URL。

将 .NET 应用程序部署到 Elastic Beanstalk

1. 在该部署工具安装的 Samples 子目录下，打开 ElasticBeanstalkDeploymentSample.txt，输入您的 AWS 访问密钥和 AWS 私有密钥，如以下示例所示。

```
### AWS Access Key and Secret Key used to create and deploy the application
 instance
AWSAccessKey = AKIAIOSFODNN7EXAMPLE
AWSSecretKey = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

> **Note**
>
> 对于 API 访问，您需要访问密钥 ID 和秘密访问密钥。使用 IAM 用户访问密钥而不是 AWS 账户根用户 访问密钥。有关创建访问密钥的更多信息，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。

2. 在命令行提示栏中，键入以下内容：

```
C:\Program Files (x86)\AWS Tools\Deployment Tool>awsdeploy.exe /w Samples
\ElasticBeanstalkDeploymentSample.txt
```

部署应用程序需要花几分钟时间。如果部署成功，您将看到消息 `Application deployment completed; environment health is Green`。

> **Note**
>
> 如果您收到以下错误，则别名记录已经存在。
>
> ```
> [Error]: Deployment to AWS Elastic Beanstalk failed with exception: DNS name
>  (MyAppEnv.elasticbeanstalk.com) is not available.
> ```
>
> 因为别名记录必须唯一，因此您需要更改 `Environment.CNAME` 中的 `ElasticBeanstalkDeploymentSample.txt`。

3. 在您的 Web 浏览器中，导航到您正在运行的应用程序的 URL。URL 的形式为 <CNAME.elasticbeanstalk.com>（例如，**`MyAppEnv.elasticbeanstalk.com`**）。

将本地 .NET 应用程序迁移到 Elastic Beanstalk

如果您考虑将 .NET 应用程序从本地服务器迁移到 Amazon Web Services (AWS)，.NET Migration Assistant for AWS Elastic Beanstalk 可能对您有用。该助手是一个交互式 PowerShell 实用程序，它将 .NET 应用程序从本地运行 IIS 的 Windows 服务器迁移到 AWS Elastic Beanstalk。该助手只需要进行少量更改或不需要更改，即可以将整个网站迁移到 Elastic Beanstalk。

有关 .NET Migration Assistant for AWS Elastic Beanstalk 及其下载的更多信息，请参阅 GitHub 上的 https://github.com/awslabs/windows-web-app-migration-assistant 存储库。

如果您的应用程序包含 Microsoft SQL Server 数据库，GitHub 上有关该助手的文档包含多个用于迁移这些数据库的选项。

将 Node.js 应用程序部署到 Elastic Beanstalk

AWS Elastic Beanstalk for Node.js 可让您使用亚马逊 Web Services 轻松部署、管理和扩展您的 Node.js Web 应用程序。适用于 Node.js 的 Elastic Beanstalk 可供任何使用 Node.js 开发或托管 Web 应用程序的人员使用。本章提供将 Node.js Web 应用程序部署到 Elastic Beanstalk 的 step-by-step 说明，并提供了数据库集成和使用 Express 框架等常见任务的演练。

部署 Elastic Beanstalk 应用程序后，您可以继续使用 EB CLI 来管理您的应用程序和环境，也可以使用 Elastic Beanstalk 控制台或 API。 AWS CLI

主题

QuickStart: 将 Node.js 应用程序部署到 Elastic Beanstalk

本 QuickStart 教程将引导您完成创建 Node.js 应用程序并将其部署到 AWS Elastic Beanstalk 环境的过程。

> **Note**
>
> 本 QuickStart 教程仅用于演示目的。请勿将本教程中创建的应用程序用于生产流量。

Sections

- 使用 Elastic Beanstalk 控制台进行部署

你的 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

如果您已经有一个 AWS 帐户，则可以继续前进先决条件 。

创建一个 AWS 账户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

先决条件

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

EB CLI

本教程使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

Node.js

按照 Node.js 网站上的 “如何安装 Node.js”，在本地计算机上安装 Node.js。

运行以下命令验证您的 Node.js 安装情况。

```
~$ node -v
```

步骤 1：创建 Node.js 应用程序

创建项目目录。

```
~$ mkdir eb-nodejs
~$ cd eb-nodejs
```

接下来，创建一个您将使用 Elastic Beanstalk 部署的应用程序。我们会创建一个“Hello World”RESTful Web 服务。

Example **~/eb-nodejs/server.js**

```
const http = require('node:http');

const hostname = '127.0.0.1';
const port = 8080;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello Elastic Beanstalk!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

此应用程序在端口 8080 上打开监听器。对于 Node.js，Elastic Beanstalk 默认通过端口 8080 将请求转发到你的应用程序。

步骤 2：在本地运行应用程序

运行以下命令以在本地运行应用程序。

```
~/eb-nodejs$ node server.js
```

您应该会看到以下文本。

```
Server running at http://127.0.0.1:8080/
```

`http://127.0.0.1:8080/`在您的网络浏览器中输入 URL 地址。浏览器应显示 “Hello Elastic Beanstalk！”。

步骤 3：使用 EB CLI 部署你的 Node.js 应用程序

运行以下命令为此应用程序创建 Elastic Beanstalk 环境。

创建环境并部署 Node.js 应用程序

1. 使用 eb init 命令，初始化 EB CLI 存储库。

   ```
   ~/eb-nodejs$ eb init -p node.js nodejs-tutorial --region us-east-2
   ```

   此命令将创建一个名为的应用程序，`nodejs-tutorial`并将您的本地存储库配置为使用最新 Node.js 平台版本创建环境。
2. （可选）再次运行 eb init 以配置默认密钥对，以便使用 SSH 连接到运行您的应用程序的 EC2 实例。

   ```
   ~/eb-nodejs$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   如果您已有密钥对，请选择一个，或按提示创建一个。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。

3. 创建环境并使用 eb create 将应用程序部署到此环境中。Elastic Beanstalk 会自动为您的应用程序生成一个 zip 文件，并将其部署到环境中的 EC2 实例。部署应用程序后，Elastic Beanstalk 在端口 8080 上启动它。

```
~/eb-nodejs$ eb create nodejs-env
```

Elastic Beanstalk 创建您的环境大约需要五分钟。

第 4 步：在 Elastic Beanstalk 上运行你的应用程序

创建环境的过程完成后，使用打开您的网站eb open。

```
~/eb-nodejs$ eb open
```

恭喜您！你已经用 Elastic Beanstalk 部署了一个 Node.js 应用程序！这将使用为应用程序创建的域名打开一个浏览器窗口。

第 5 步：清理

完成应用程序的使用后，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源。

要使用 EB CLI 终止您的 Elastic Beanstalk 环境，请运行以下命令。

```
~/eb-nodejs$ eb terminate
```

AWS 您的应用程序的资源

您刚刚创建了一个单实例应用程序。它可用作带有单个 EC2 实例的简单示例应用程序，因此不需要负载平衡或 auto Scaling。对于单实例应用程序，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

  各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。

- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 管理所有这些资源。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。您还可以使用 Elastic Beanstalk 控制台探索您的新环境。有关详细步骤，请参阅本指南 “入门” 一章中的 “探索您的环境”。

> **尝试更多教程**
>
> 如果您想尝试其他包含不同示例应用程序的教程，请参阅Node.js 的更多示例应用程序和教程。

部署一两个示例应用程序并准备好开始在本地开发和运行 Node.js 应用程序之后，请参阅设置 Node.js 开发环境。

使用 Elastic Beanstalk 控制台进行部署

您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。

设置 Node.js 开发环境

设置 Node.js 开发环境以在本地测试应用程序，然后再将之部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

有关适用于所有语言的常见设置步骤和工具，请参阅配置您的开发计算机。

主题

- 安装 Node.js。
- 确认 npm 安装
- 安装适用于 Node.js 的AWS开发工具包
- 安装 Express 生成器
- 设置 Express 框架和服务器

安装 Node.js。

安装 Node.js 以在本地运行 Node.js 应用程序。如果您没有特别的要求，请获取 Elastic Beanstalk 支持的最新版本。有关受支持版本的列表，请参阅 AWS Elastic Beanstalk 平台文档中的 Node.js。

请从 nodejs.org 下载 Node.js。

确认 npm 安装

Node.js 使用 npm 程序包管理器帮助您安装要在应用程序中使用的工具和框架。由于 npm 是随 Node.js 一起分发的，因此在下载并安装 Node.js 时将自动安装 npm。要确认是否已安装 npm，可以运行以下命令：

```
$ npm -v
```

有关 npm 的更多信息，请访问 npmjs 网站。

安装适用于 Node.js 的AWS开发工具包

如果您需要在应用程序中管理 AWS 资源，请安装 AWS SDK for JavaScript in Node.js。使用 npm 安装开发工具包：

```
$ npm install aws-sdk
```

有关更多信息，请访问 AWS SDK for JavaScript in Node.js 主页。

安装 Express 生成器

Express 是运行在 Node.js 上的 Web 应用程序框架。要使用它，请先安装 Express 生成器命令行应用程序。安装 Express 生成器后，您可以运行 express 命令为 Web 应用程序生成基础项目结构。安装基础项目、文件和依赖项后，就可以在开发计算机上启动本地 Express 服务器。

> **Note**
>
> - 以下步骤向您演示了如何在 Linux 操作系统上安装 Express 生成器。
> - 对于 Linux，根据您对系统目录的权限级别，可能需要为一些命令添加前缀 sudo。

要在您的开发环境中安装 Express 生成器

1. 为您的 Express 框架和服务器创建工作目录。

   ```
   ~$ mkdir node-express
   ~$ cd node-express
   ```

2. 全局安装 Express，以便您拥有 `express` 命令的访问权限。

   ```
   ~/node-express$ npm install -g express-generator
   ```

3. 根据操作系统，您可能需要设置路径才能运行 `express` 命令。如果您需要设置路径变量，则上一步的输出会提供信息。以下是 Linux 的示例。

   ```
   ~/node-express$ export PATH=$PATH:/usr/local/share/npm/bin/express
   ```

   当您按照本章中的教程进行操作时，需要从不同的目录运行 express 命令。每个教程都在其自己的目录中设置了基本的 Express 项目结构。

现在，您已经安装了 Express 命令行生成器。您可以使用它为 Web 应用程序创建框架目录、设置依赖项和启动 Web 应用程序服务器。接下来，将在创建的 `node-express` 目录中逐步执行完成此操作的步骤。

设置 Express 框架和服务器

按照以下步骤创建基本 Express 框架目录和内容。本章中的教程还包括在教程的每个应用程序目录中设置基础 Express 框架的步骤。

要设置 Express 框架和服务器

1. 运行 `express` 命令。这将生成 `package.json`、`app.js`，以及几个目录。

```
~/node-express$ express
```

在系统提示您是否要继续时，键入 **y**。

2. 设置本地依赖项。

```
~/node-express$ npm install
```

3. 验证 Web 应用程序服务器是否已启动。

```
~/node-express$ npm start
```

您应该可以看到类似于如下所示的输出内容：

```
> nodejs@0.0.0 start /home/local/user/node-express
> node ./bin/www
```

默认情况下，服务器在端口 3000 上运行。要测试，请在另一个终端中运行 `curl http://localhost:3000`，或在本地计算机上打开浏览器并输入 URL 地址 `http://localhost:3000`。

按 Ctrl+C 以停止该服务器。

使用 Elastic Beanstalk Node.js 平台

AWS Elastic Beanstalk Node.js 平台是一组平台版本，适用于可在 NGINX 代理服务器后方运行的 Node.js Web 应用程序。

Elastic Beanstalk 提供了配置选项，可供您用于自定义在 Elastic Beanstalk 环境中的 EC2 实例上运行的软件。您可配置应用程序所需的环境变量，启用到 Amazon S3 的日志轮换，并将应用程序源中包含静态文件的文件夹映射至代理服务器所提供的路径。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

您可以在源包中包含一个 `Package.json` 文件，以便在部署期间安装软件包、提供 start 命令以及指定您希望应用程序使用的 Node.js 版本。您可以包含一个 `npm-shrinkwrap.json` 文件来锁定依赖项版本。

Node.js 平台包含一个代理服务器，以服务静态资产、将流量转发到应用程序和压缩响应。您可以扩展或覆盖默认代理配置，以适应高级方案。

可提供多个选项供您启动应用程序。您可以将 Procfile 添加到源包来指定用于启动应用程序的命令。如果您不提供 `Procfile`，但提供 `package.json` 文件，则 Elastic Beanstalk 将运行 `npm start`。如果您未提供上述任一项，则 Elastic Beanstalk 会按此顺序查找 `app.js` 或 `server.js` 文件并运行该脚本。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅 the section called “扩展 Linux 平台”。

配置您的 Node.js 环境

您可以使用 Node.js 平台设置来微调 Amazon EC2 实例的行为。您可以使用 Elastic Beanstalk 控制台编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，并配置应用程序可从环境中读取的变量。

在 Elastic Beanstalk 控制台中配置 Node.js 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

容器选项

您可以指定以下特定于平台的选项：

- Proxy server（代理服务器）– 要在环境实例上使用的代理服务器。默认使用 NGNIX。

日志选项

日志选项部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

环境属性

使用 Environment Properties（环境属性）部分，在运行应用程序的 Amazon EC2 实例上指定环境配置设置。这些设置会以密钥值对的方式传递到应用程序。

在运行于 AWS Elastic Beanstalk 中的 Node.js 环境内，您可以通过运行 `process.env.ENV_VARIABLE` 来访问环境变量。

```
var endpoint = process.env.API_ENDPOINT
```

Node.js 平台会将 PORT 环境变量设置为代理服务器将流量传输到的端口。有关更多信息，请参阅配置代理服务器。

参阅 环境属性和其他软件设置 了解更多信息。

配置 Amazon Linux AMI（在 Amazon Linux 2 之前）Node.js 环境

仅在使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前）的 Elastic Beanstalk Node.js 环境中支持以下控制台软件配置类别。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

容器选项 — Amazon Linux AMI (AL1)

在配置页面上，指定以下内容：

- Proxy server（代理服务器）– 指定要用来代理与 Node.js 连接的 Web 服务器。默认使用 NGINX。如果选择 none，则静态文件映射不会生效，并且 GZIP 压缩会被禁用。
- Node.js 版本 – 指定 Node.js 的版本。有关受支持的 Node.js 版本的列表，请参阅 AWS Elastic Beanstalk 平台指南中的 Node.js。
- GZIP compression（Gzip 压缩）– 指定是否启用 GZIP 压缩。默认情况下已启用 GZIP 压缩。
- Node command（节点命令）– 供您输入用于启动 Node.js 应用程序的命令。空字符串（默认值）表示 Elastic Beanstalk 将依次使用 `app.js`、`server.js` 和 `npm start`。

Node.js 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间中。

您可以使用 `aws:elasticbeanstalk:environment:proxy` 命名空间选择要在环境的实例上使用的代理。以下示例将您的环境配置为使用 Apache HTTPD 代理服务器。

Example .ebextensions/nodejs-settings.config

```
option_settings:
```

```
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
```

您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间将代理配置为提供静态文件。有关更多信息以及示例，请参阅 the section called “静态文件”。

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

Amazon Linux AMI（在 Amazon Linux 2 之前）Node.js 平台

如果您的 Elastic Beanstalk Node.js 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请考虑本节中的特定配置和建议。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

Node.js 平台特定的配置选项 — Amazon Linux AMI (AL1)

Elastic Beanstalk 支持用于 Amazon Linux AMI Node.js 平台版本的一些特定于平台的配置选项。可以选择先于应用程序运行的代理服务器，选择要运行的特定版本的 Node.js，并选择用于运行应用程序的命令。

对于代理服务器，您可以使用 NGINX 或 Apache 代理服务器。您可以将 none 值设置为 `ProxyServer` 选项。在这种设置下，Elastic Beanstalk 将您的应用程序作为独立项运行，而不是在任何代理服务器后面运行。如果您的环境运行独立应用程序，请更新您的代码以侦听 NGINX 将流量转发到的端口。

```
var port = process.env.PORT || 8080;

app.listen(port, function() {
```

```
  console.log('Server running at http://127.0.0.1:%s', port);
});
```

Node.js 语言版本 — Amazon Linux AMI (AL1)

在支持的语言版本方面，Node.js Amazon Linux AMI 平台与其他 Elastic Beanstalk 托管平台略有不同。这是因为每个 Node.js 平台版本均仅支持几个 Node.js 语言版本。有关受支持的 Node.js 版本的列表，请参阅 AWS Elastic Beanstalk 平台指南中的 Node.js。

您可以使用特定于平台的配置选项来设置语言版本。有关说明，请参阅 the section called “配置您的 Node.js 环境”。或者，使用 Elastic Beanstalk 控制台在更新平台版本的过程中，更新您的环境使用的 Node.js 版本。

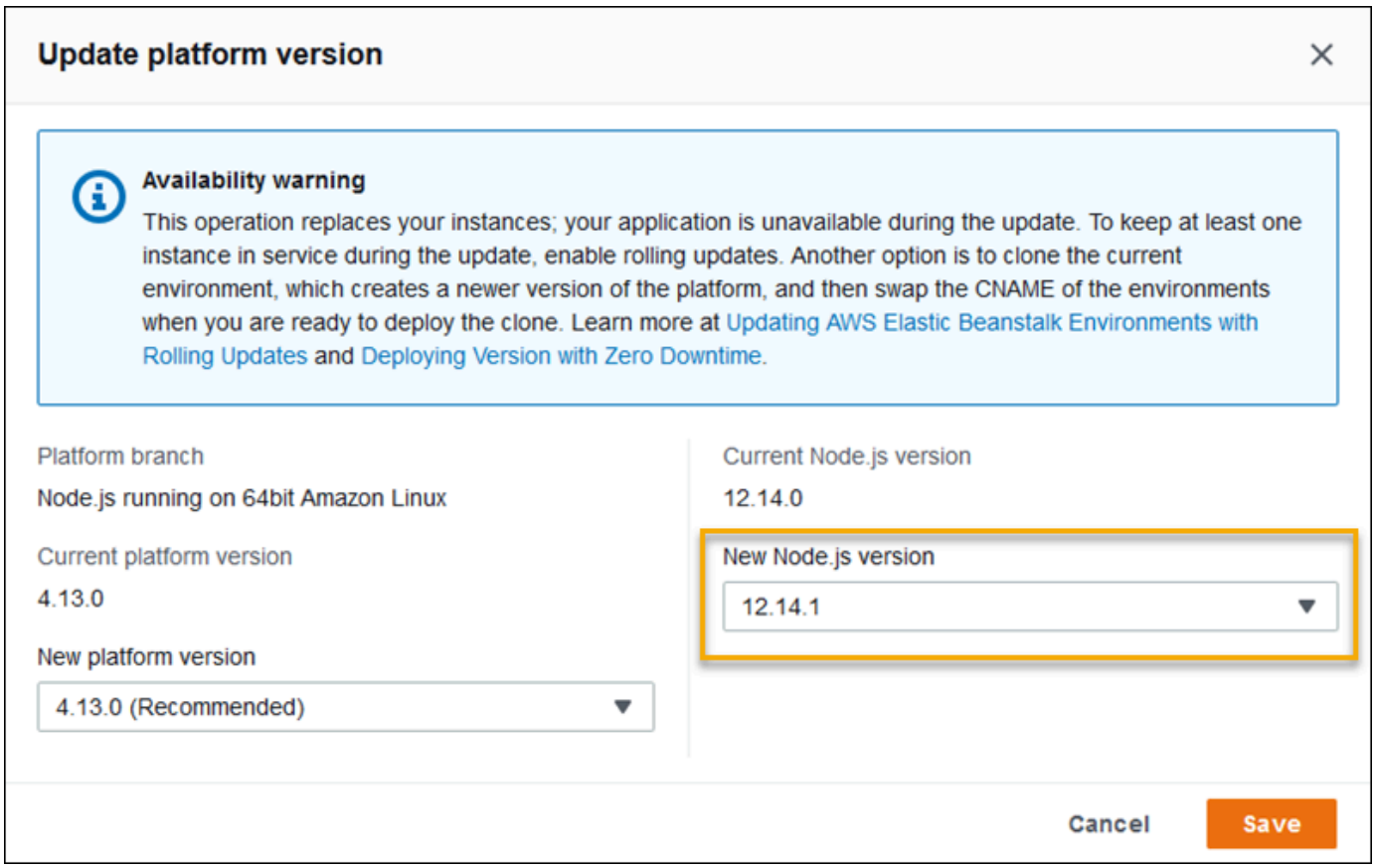
> **Note**
>
> 如果对您正在使用的 Node.js 版本的支持已从平台中移除，则您必须先更改或移除版本设置再进行平台更新。当在一个或多个 Node.js 版本中识别到安全漏洞时，可能会发生这种情况。发生此情况时，尝试更新到不支持所配置的 NodeVersion 的新平台版本会失败。为避免需要创建新环境，请将 NodeVersion 配置选项更改为旧平台版本和新平台版本均支持的 Node.js 版本，或移除选项设置，然后执行平台更新。

在 Elastic Beanstalk 控制台中配置环境的 Node.js 版本

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上的 Platform（平台）下，选择 Change（更改）。
4. 在 Update platform version（更新平台版本）对话框中，选择 Node.js 版本。



5. 选择 Save（保存）。

Node.js 配置命名空间 — Amazon Linux AMI (AL1)

Node.js Amazon Linux AMI 平台在 `aws:elasticbeanstalk:container:nodejs:staticfiles` 和 `aws:elasticbeanstalk:container:nodejs` 命名空间中定义附加选项。

以下配置文件将指示 Elastic Beanstalk 使用 `npm start` 运行应用程序。它也将代理类型设置为 Apache 并启用压缩。最后，它将代理配置为从两个源目录提供静态文件。一个源是来自 `statichtml` 源目录的网站根目录下的 `html` 路径的 HTML 文件。另一个源是来自 `staticimages` 源目录的网站根目录下的 `images` 路径的 HTML 文件。

Example .ebextensions/node-settings.config

```
option_settings:
  aws:elasticbeanstalk:container:nodejs:
    NodeCommand: "npm start"
    ProxyServer: apache
    GzipCompression: true
```

```
  aws:elasticbeanstalk:container:nodejs:staticfiles:
    /html: statichtml
    /images: staticimages
```

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

使用 Procfile 配置应用程序进程

您可以在源包的根目录中包含一个名为 `Procfile` 的文件，来指定启动应用程序的命令。

Example Procfile

```
web: node index.js
```

有关 `Procfile` 用法的信息，请展开 the section called “扩展 Linux 平台” 中的 Buildfile 和 Procfile 部分。

> **Note**
>
> 此功能将替换 `aws:elasticbeanstalk:container:nodejs` 命名空间中的旧 `NodeCommand` 选项。

配置您的应用程序的依赖项

您的应用程序可能有某些 Node.js 模块的依赖项，例如您在 `require()` 语句中指定的项。这些模块存储在 `node_modules` 目录中。当您的应用程序运行时，Node.js 将从此目录加载模块。有关更多信息，请参阅 Node.js 文档中的从 node_modules 文件夹加载。

您可以使用 `package.json` 文件指定这些模块依赖项。如果 Elastic Beanstalk 检测到此文件，但 `node_modules` 目录不存在，则 Elastic Beanstalk 将以 webapp 用户身份运行 `npm install`。`npm install` 命令将依赖项安装在 Elastic Beanstalk 事先创建的 `node_modules` 目录中。`npm install` 命令会从公共 npm 注册表或其他位置访问 `package.json` 文件中列出的软件包。有关更多信息，请参阅 npm Docs 网站。

如果 Elastic Beanstalk 检测到 `node_modules` 目录，则即使存在 `package.json` 文件，Elastic Beanstalk 也不会运行 `npm install`。Elastic Beanstalk 假设依赖项包在 `node_modules` 目录中可供 Node.js 访问和加载。

以下各部分提供有关为应用程序建立 Node.js 模块依赖项的更多信息。

> **Note**
>
> 如果您在 Elastic Beanstalk 运行 `npm install` 时遇到任何部署问题，请考虑其他方法。在您的应用程序源包中包括含依赖模块的 `node_modules` 目录。这样做可以避免在调查问题时从公共 npm 注册表安装依赖项时出现问题。由于依赖模块来自本地目录，这样做也可能有助于缩短部署时间。有关更多信息，请参阅 在 node_modules 目录中包括 Node.js 依赖项。

使用 package.json 文件指定 Node.js 依赖项

包含项目源的根目录中的 `package.json` 文件来指定依赖项包并提供启动命令。当 `package.json` 文件存在，且项目源的根目录中不存在 `node_modules` 目录时，Elastic Beanstalk 会以 webapp 用户的身份运行 `npm install` 来从公共 npm 注册表安装依赖项。Elastic Beanstalk 还使用 `start` 命令来启动您的应用程序。有关 `package.json` 文件的更多信息，请参阅 npm Docs 网站中的在 `package.json` 文件中指定依赖项。

使用 `scripts` 关键字来提供启动命令。当前使用 `scripts` 关键字，而不是 `aws:elasticbeanstalk:container:nodejs` 命名空间中的原有 `NodeCommand` 选项。

Example package.json - Express

```
{
    "name": "my-app",
    "version": "0.0.1",
    "private": true,
    "dependencies": {
      "ejs": "latest",
      "aws-sdk": "latest",
      "express": "latest",
      "body-parser": "latest"
    },
    "scripts": {
      "start": "node app.js"
    }
  }
```

生产模式和开发依赖项

要在 `package.json` 文件中指定依赖项，请使用 dependencies 和 devDependencies 属性。dependencies 属性指定您的应用程序在生产中所需的软件包。devDependencies 属性指定仅本地开发和测试需要的软件包。

Elastic Beanstalk 使用以下命令以 webapp 用户身份运行 `npm install`。命令选项因运行应用程序的平台分支中包含的 npm 版本而异。

- npm v6 – 默认情况下，Elastic Beanstalk 在生产模式下安装依赖项。它使用命令 `npm install --production`。
- npm v7 或更高版本 – Elastic Beanstalk 省略了 devDependencies。它使用命令 `npm install --omit=dev`。

上面列出的两个命令都没有安装属于 devDependencies 的软件包。

如果您需要安装 devDependencies 软件包，则请将 NPM_USE_PRODUCTION 环境属性设置为 `false`。使用此设置，在运行 npm install 时将不使用上述选项。这将导致 devDependencies 软件包被安装。

SSH 和 HTTPS

从 2023 年 3 月 7 日的 Amazon Linux 2 平台版本开始，您还可以使用 SSH 和 HTTPS 协议从 Git 存储库检索软件包。平台分支 Node.js 16 同时支持 SSH 和 HTTPS 协议。Node.js 14 仅支持 HTTPS 协议。

Example package.json – Node.js 16 同时支持 HTTPS 和 SSH

```
...
"dependencies": {
  "aws-sdk": "https://github.com/aws/aws-sdk-js.git",
  "aws-chime": "git+ssh://git@github.com:aws/amazon-chime-sdk-js.git"
}
```

版本和版本范围

> **Important**
>
> 指定版本范围的功能不适用于在 AL2023 上运行的 Node.js 平台分支。在 AL2023 上的特定 Node.js 分支中，只支持一个 Node.js 版本。如果您的 `package.json` 文件指定了版本范围，我们将忽略它并默认设为 Node.js 的平台分支版本。

使用 `package.json` 文件中的 `engines` 关键字指定您希望应用程序使用的 Node.js 版本。您还可以使用 npm 表示法指定版本范围。有关版本范围的语法的更多信息，请参阅 Node.js 网站上的使用 npm 的语义版本控制。Node.js `package.json` 文件中的 `engines` 关键字取代 `aws:elasticbeanstalk:container:nodejs` 命名空间中的原有 `NodeVersion` 选项。

Example **`package.json`** – 单一 Node.js 版本

```
{
    ...
    "engines": { "node" : "14.16.0" }
  }
```

Example **`package.json`** – Node.js 版本范围

```
{
    ...
    "engines": { "node" : ">=10 <11" }
  }
```

当指示版本范围时，Elastic Beanstalk 会安装该平台在该范围内可用的最新 Node.js 版本。在此示例中，范围指示版本必须大于或等于版本 10，但小于版本 11。因此，Elastic Beanstalk 安装了最新的 Node.js 版本 10.x.y，该版本在支持的平台上可用。

请注意，您只能指定与平台分支对应的 Node.js 版本。例如，如果您使用的是 Node.js 16 平台分支，则只能指定 16.x.y Node.js 版本。您可以使用 npm 支持的版本范围选项来提高灵活度。有关每个平台分支的有效 Node.js 版本，请参阅 AWS Elastic Beanstalk 平台指南中的 Node.js。

> **Note**
>
> 如果对您正在使用的 Node.js 版本的支持已从平台中移除，则您必须先更改或移除 Node.js 版本设置再进行平台更新。当在一个或多个 Node.js 版本中识别到安全漏洞时，可能会发生这种情况。
>
> 发生此情况时，尝试更新到不支持所配置的 Node.js 版本的新平台版本会失败。为避免需要创建新环境，请将 `package.json` 中的 Node.js 版本设置更改为旧平台版本和新平台版本都支持的 Node.js 版本。您可以选择指定包含受支持版本的 Node.js 版本范围，如本主题前面所述。您还可以选择删除设置，然后部署新的源包。

在 node_modules 目录中包括 Node.js 依赖项

要将依赖项程序包与应用程序代码一起部署到环境实例，请将它们包括在项目源根目录下名为 `node_modules` 的目录中。有关更多信息，请参阅 npm Docs 网站中的在本地下载和安装软件包。

将 `node_modules` 目录部署到 Amazon Linux 2 Node.js 平台版本时，Elastic Beanstalk 假定您提供了自己的依赖项程序包，并避免安装 package.json 文件中指定的依赖项。Node.js 在 `node_modules` 目录中查找依赖项。有关详细信息，请参阅 Node.js 文档中的从 node_modules 文件夹加载。

> **Note**
>
> 如果您在 Elastic Beanstalk 运行 `npm install` 时遇到任何部署问题，请在调查问题时考虑使用本主题中描述的方法作为解决方法。

使用 npm shrinkwrap 锁定依赖项

Node.js 平台将在您每次部署时以 webapp 用户身份运行 `npm install`。新版本的依赖项可用时，如果您部署应用程序，就会安装这些依赖项，这可能会导致部署需要较长时间才能完成。

您可以通过创建将应用程序的依赖项锁定为当前版本的 `npm-shrinkwrap.json` 文件来避免升级依赖项。

```
$ npm install
$ npm shrinkwrap
wrote npm-shrinkwrap.json
```

在源包中包含此文件可确保依赖项仅安装一次。

配置代理服务器

Elastic Beanstalk 可使用 NGINX 或 Apache HTTPD 作为反向代理，将应用程序映射到端口 80 上的 Elastic Load Balancing 负载均衡器。默认为 NGINX。Elastic Beanstalk 提供一个默认代理配置，您可以扩展该配置，也可以使用您自己的配置完全覆盖该配置。

默认情况下，Elastic Beanstalk 将代理配置为通过端口 5000 向您的应用程序转发请求。您可以覆盖默认端口，方法是将 `PORT` 环境属性设置为主应用程序侦听的端口。

> **Note**
>
> 应用程序侦听的端口不会影响 NGINX 服务器为了从负载均衡器接收请求而侦听的端口。

在平台版本上配置代理服务器

所有 AL2023/AL2 平台都支持统一的代理配置功能。有关在运行 AL2023/AL2 的平台版本上配置代理服务器的更多信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

在 Amazon Linux AMI（在 Amazon Linux 2 之前）上配置代理

如果您的 Elastic Beanstalk Node.js 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读本节中的信息。

> **注意**
>
> - 本主题中的信息仅适用于基于 Amazon Linux AMI (AL1) 的平台分支。AL2023/AL2 平台分支与以前的 Amazon Linux AMI（AL1）平台版本不兼容，需要不同的配置设置。
> - 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

扩展并覆盖默认代理配置 — Amazon Linux AMI（AL1）

Node.js 平台使用反向代理将来自实例上的端口 80 的请求中继到在端口 8081 上侦听的应用程序。Elastic Beanstalk 提供一个默认代理配置，您可以扩展该配置，也可以使用您自己的配置完全覆盖该配置。

要扩展默认配置，请使用配置文件将 `.conf` 文件添加到 `/etc/nginx/conf.d`。有关具体示例，请参阅 在运行 Node.js 的 EC2 实例上终止 HTTPS。

Node.js 平台会将 PORT 环境变量设置为代理服务器将流量传输到的端口。在代码中读取此变量可配置应用程序的端口。

```
var port = process.env.PORT || 3000;
```

```
    var server = app.listen(port, function () {
        console.log('Server running at http://127.0.0.1:' + port + '/');
    });
```

默认 NGINX 配置会将流量转发到 127.0.0.1:8081 上名为 nodejs 的上游服务器。可以删除默认配置并在配置文件中提供您自己的配置。

Example .ebextensions/proxy.config

以下示例将删除默认配置并添加一个将流量转发到端口 5000 (而不是 8081) 的自定义配置。

```
files:
  /etc/nginx/conf.d/proxy.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      upstream nodejs {
        server 127.0.0.1:5000;
        keepalive 256;
      }

      server {
        listen 8080;

        if ($time_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {
            set $year $1;
            set $month $2;
            set $day $3;
            set $hour $4;
        }
        access_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour
 healthd;
        access_log  /var/log/nginx/access.log  main;

        location / {
            proxy_pass  http://nodejs;
            proxy_set_header   Connection "";
            proxy_http_version 1.1;
            proxy_set_header        Host            $host;
            proxy_set_header        X-Real-IP       $remote_addr;
            proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        }
```

```
        gzip on;
        gzip_comp_level 4;
        gzip_types text/html text/plain text/css application/json application/x-
javascript text/xml application/xml application/xml+rss text/javascript;

        location /static {
            alias /var/app/current/static;
        }

      }

  /opt/elasticbeanstalk/hooks/configdeploy/post/99_kill_default_nginx.sh:
    mode: "000755"
    owner: root
    group: root
    content: |
      #!/bin/bash -xe
      rm -f /etc/nginx/conf.d/00_elastic_beanstalk_proxy.conf
      service nginx stop
      service nginx start

container_commands:
  removeconfig:
    command: "rm -f /tmp/deployment/config/
#etc#nginx#conf.d#00_elastic_beanstalk_proxy.conf /etc/nginx/
conf.d/00_elastic_beanstalk_proxy.conf"
```

示例配置（`/etc/nginx/conf.d/proxy.conf`）使用 `/etc/nginx/conf.d/00_elastic_beanstalk_proxy.conf` 中的默认配置作为包括具有压缩和日志设置以及静态文件映射的默认服务器数据块的基础。

`removeconfig` 命令删除容器的默认配置，以确保代理服务器使用自定义配置。Elastic Beanstalk 在每次配置部署期间都会重新创建默认配置。考虑到这一点，在下列示例中，添加了一个配置后部署挂钩 (`/opt/elasticbeanstalk/hooks/configdeploy/post/99_kill_default_nginx.sh`)。这删除了默认配置并重新启动代理服务器。

> **Note**
>
> 在 Node.js 平台将来的版本中，默认配置可能会发生更改。请使用该配置的最新版本作为您自定义的基础，以确保兼容性。

如果要覆盖默认配置，您必须定义任何静态文件映射和 GZIP 压缩。这是因为平台无法应用标准设置。

Node.js 的更多示例应用程序和教程

要开始启用 Node.js 应用程序 AWS Elastic Beanstalk，您只需要一个应用程序源包，将其作为第一个应用程序版本上传并部署到环境中。本QuickStart 对于 Node.js主题将引导您完成使用 EB CLI 启动示例 Node.js 应用程序的过程。本节提供了其他应用程序和教程。

使用示例 Node.js 应用程序启动环境

Elastic Beanstalk 为每个平台提供了单页示例应用程序以及更复杂的示例，这些示例展示了 AWS 其他资源的使用情况，例如 Amazon RDS 以及特定于语言或平台的功能和 API。

> **Note**
>
> 按照源包 `README.md` 文件中的步骤进行部署。

样本

环境类型	源代码包	描述
Web 服务器	nodejs.zip	单页面应用程序。 要使用 EB CLI 启动示例应用程序，请参阅QuickStart 对于 Node.js。 您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。
带 Amazon RDS 的 Web Server	nodejs-example-express-rds.zip	使用 Express 框架和 Amazon Relational Database Service（RDS）的 Hiking 日志应用程序。 教程
Amazon 网络服务器 ElastiCache	nodejs-example-	使用 Amazon ElastiCache 进行集群的 Express Web 应用程序。集群功能增强了 Web 应用程序的高可用性、性能和安全性。

环境类型	源代码包	描述	
	express-elasticache.zip	教程	
带 DynamoDB、Amazon SNS 和 Amazon SQS 的 Web Server	nodejs-example-dynamo.zip	为新公司的市场营销活动收集用户联系信息的 Express 网站。使用 Node.js JavaScript 中的 AWS 软件开发工具包向 DynamoDB 表写入条目，使用 Elastic Beanstalk 配置文件在 DynamoDB、Amazon SNS 和亚马逊 SQS 中创建资源。 教程	

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或完全不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。有关应用程序部署的详细信息，请参阅部署应用程序的新版本。

在部署了一两个示例应用程序并准备好开始在本地开发和运行 Node.js 应用程序之后，请参阅设置 Node.js 开发环境使用所需的所有工具设置 Node.js 开发环境。

将 Express 应用程序部署到 Elastic Beanstalk

本部分演示如何使用 Elastic Beanstalk 命令行界面（EB CLI）向 Elastic Beanstalk 部署示例应用程序，然后更新该应用程序以使用 Express 框架。

先决条件

本教程需要以下先决条件：

- Node.js 运行时
- 默认 Node.js 程序包管理器软件 npm
- Express 命令行生成器
- Elastic Beanstalk 命令行界面 (EB CLI)

有关安装列出的前三个组件和设置本地开发环境的详细信息，请参阅 设置 Node.js 开发环境。在本教程中，您无需安装 AWS 适用于 Node.js 的 SDK，参考主题中也提到了这一点。

有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

创建 Elastic Beanstalk 环境

您的应用程序目录

本教程为应用程序源包使用名为 `nodejs-example-express-rds` 的目录。为本教程创建 `nodejs-example-express-rds` 目录。

```
~$ mkdir nodejs-example-express-rds
```

> **Note**
>
> 本章中的每个教程都为应用程序源包使用自己的目录。该目录名称与教程使用的示例应用程序的名称相匹配。

将您当前的工作目录更改为 `nodejs-example-express-rds`。

```
~$ cd nodejs-example-express-rds
```

现在，让我们设置运行 Node.js 平台和示例应用程序的 Elastic Beanstalk 环境。我们将使用 Elastic Beanstalk 命令行界面（EB CLI）。

要为您的应用程序配置 EB CLI 存储库，并创建运行 Node.js 平台的 Elastic Beanstalk 环境

1. 使用 eb init 命令创建存储库。

   ```
   ~/nodejs-example-express-rds$ eb init --platform node.js --region <region>
   ```

   此命令在名为 `.elasticbeanstalk` 的文件夹中创建配置文件，该配置文件指定用于为您的应用程序创建环境的设置；并创建以当前文件夹命名的 Elastic Beanstalk 应用程序。
2. 使用 eb create 命令创建运行示例应用程序的环境。

   ```
   ~/nodejs-example-express-rds$ eb create --sample nodejs-example-express-rds
   ```

此命令使用 Node.js 平台的默认设置以及以下资源来创建负载均衡环境：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

  > **Note**
  >
  > 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

3. 当环境创建完成后，使用 eb open 命令在默认浏览器中打开环境 URL。

```
~/nodejs-example-express-rds$ eb open
```

您现在已经使用示例应用程序创建了 Node.js Elastic Beanstalk 环境。您可以使用自己的应用程序对其进行更新。接下来，我们会更新示例应用程序，以使用 Express 框架。

更新应用程序以使用 Express

在创建具有示例应用程序的环境后，可将其更新为自己的应用程序。在此过程中，首先运行 express 和 npm install 命令，以在您的应用程序目录中设置 Express 框架。然后，使用 EB CLI 通过更新后的应用程序更新您的 Elastic Beanstalk 环境。

更新您的应用程序以使用 Express

1. 运行 `express` 命令。这将生成 `package.json`、`app.js`，以及几个目录。

   ```
   ~/nodejs-example-express-rds$ express
   ```

   在系统提示您是否要继续时，键入 **y**。

   > **Note**
   >
   > 如果 express 命令不起作用，则您可能没有按照前面的先决条件部分所述安装 Express 命令行生成器。或者，可能需要设置本地计算机的目录路径设置才能运行 express 命令。有关设置开发环境的详细步骤，请参阅先决条件部分，以便您可以继续学习本教程。

2. 设置本地依赖项。

   ```
   ~/nodejs-example-express-rds$ npm install
   ```

3. （可选）验证 Web 应用程序服务器已启动。

   ```
   ~/nodejs-example-express-rds$ npm start
   ```

   您应该可以看到类似于如下所示的输出内容：

   ```
   > nodejs@0.0.0 start /home/local/user/node-express
   > node ./bin/www
   ```

   默认情况下，服务器在端口 3000 上运行。要测试，请在另一个终端中运行 `curl http://localhost:3000`，或在本地计算机上打开浏览器并输入 URL 地址 `http://localhost:3000`。

按 Ctrl+C 以停止该服务器。

4. 使用 eb deploy 命令将更改部署到您的 Elastic Beanstalk 环境。

```
~/nodejs-example-express-rds$ eb deploy
```

5. 在环境变为绿色并准备就绪后，刷新 URL 以验证环境是否工作。您应看到一个显示 Welcome to Express（欢迎使用 Express）的网页。

接下来，让我们更新 Express 应用程序以使用静态文件并添加新页面。

配置静态文件并向 Express 应用程序添加新页面

1. 添加包含以下内容的 .ebextensions 文件夹中的第二个配置文件：

   nodejs-example-express-rds/.ebextensions/staticfiles.config

```
option_settings:
    aws:elasticbeanstalk:environment:proxy:staticfiles:
        /stylesheets: public/stylesheets
```

   此设置将代理服务器配置为从应用程序的 public 路径上的 /public 文件夹中提供文件。从代理服务器静态提供文件可减少应用程序的负载。有关更多信息，请参阅本章前面的静态文件。

2. （可选）要确认静态映射配置正确，请注释掉 nodejs-example-express-rds/app.js 中的静态映射配置。这将从节点应用程序中删除映射。

```
//  app.use(express.static(path.join(__dirname, 'public')));
```

   即使在将此行注释掉之后，上一步的 staticfiles.config 文件中的静态文件映射仍应成功加载样式表。要验证静态文件映射是通过代理静态文件配置而不是快速应用程序加载的，请删除 option_settings: 后的值。将其从静态文件配置和节点应用程序中删除后，样式表将无法加载。

   测试完成后，记得重置 nodejs-example-express-rds/app.js 和 staticfiles.config 的内容。

3. 添加 nodejs-example-express-rds/routes/hike.js。键入以下内容：

```
exports.index = function(req, res) {
```

```
 res.render('hike', {title: 'My Hiking Log'});
};

exports.add_hike = function(req, res) {
};
```

4. 更新 nodejs-example-express-rds/app.js 以包含三个新行。

   首先，添加以下行来为此路由添加 require：

   ```
   var hike = require('./routes/hike');
   ```

   您的文件应类似于以下代码段：

   ```
   var express = require('express');
   var path = require('path');
   var hike = require('./routes/hike');
   ```

   然后，将以下两行添加到 nodejs-example-express-rds/app.js 中的 var app = express(); 后面

   ```
   app.get('/hikes', hike.index);
   app.post('/add_hike', hike.add_hike);
   ```

   您的文件应类似于以下代码段：

   ```
   var app = express();
   app.get('/hikes', hike.index);
   app.post('/add_hike', hike.add_hike);
   ```

5. 将 nodejs-example-express-rds/views/index.jade 复制到 nodejs-example-express-rds/views/hike.jade。

   ```
   ~/nodejs-example-express-rds$ cp views/index.jade views/hike.jade
   ```

6. 使用 eb deploy 命令部署更改。

   ```
   ~/nodejs-example-express-rds$ eb deploy
   ```

7. 您的环境将在几分钟后进行更新。在环境变为绿色并准备就绪后，通过刷新浏览器并将 **hikes** 追加到 URL 末尾（例如 `http://node-express-env-syypntcz2q.elasticbeanstalk.com/hikes`），验证环境是否正常工作。

   您应看到标题为 My Hiking Log 的网页。

现在，您已经创建使用 Express 框架的 Web 应用程序。在下一节中，我们将修改应用程序以使用 Amazon Relational Database Service（RDS）来存储 Hiking 日志。

更新应用程序以使用 Amazon RDS

在下一步中，我们将应用程序更新为使用 Amazon RDS for MySQL。

要更新您的应用程序以使用 RDS for MySQL

1. 要创建耦合到您的 Elastic Beanstalk 环境的 RDS for MySQL 数据库，请按照本章后面所包含的[添加数据库]主题中的说明进行操作。添加一个数据库实例大约需要 10 分钟。
2. 使用以下内容更新 `package.json` 中的依赖项部分：

```
"dependencies": {
    "async": "^3.2.4",
    "express": "4.18.2",
    "jade": "1.11.0",
    "mysql": "2.18.1",
    "node-uuid": "^1.4.8",
    "body-parser": "^1.20.1",
    "method-override": "^3.0.0",
    "morgan": "^1.10.0",
    "errorhandler": "^1.5.1"
  }
```

3. 运行 npm install。

```
~/nodejs-example-express-rds$ npm install
```

4. 更新 `app.js` 以连接到数据库、创建表并插入单个默认 Hiking 日志。每次部署此应用程序时，它都会删除之前的 hikes 表并重新创建它。

```
/**
 * Module dependencies.
 */
```

```
 const express = require('express')
 , routes = require('./routes')
 , hike = require('./routes/hike')
 , http = require('http')
 , path = require('path')
 , mysql = require('mysql')
 , async = require('async')
 , bodyParser = require('body-parser')
 , methodOverride = require('method-override')
 , morgan = require('morgan')
 , errorhandler = require('errorhandler');

const { connect } = require('http2');

const app = express()

app.set('views', __dirname + '/views')
app.set('view engine', 'jade')
app.use(methodOverride())
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
app.use(express.static(path.join(__dirname, 'public')))


app.set('connection', mysql.createConnection({
host: process.env.RDS_HOSTNAME,
user: process.env.RDS_USERNAME,
password: process.env.RDS_PASSWORD,
port: process.env.RDS_PORT}));

function init() {
 app.get('/', routes.index);
 app.get('/hikes', hike.index);
 app.post('/add_hike', hike.add_hike);
}

const client = app.get('connection');
async.series([
 function connect(callback) {
   client.connect(callback);
   console.log('Connected!');
 },
 function clear(callback) {
```

```
    client.query('DROP DATABASE IF EXISTS mynode_db', callback);
  },
  function create_db(callback) {
    client.query('CREATE DATABASE mynode_db', callback);
  },
  function use_db(callback) {
    client.query('USE mynode_db', callback);
  },
  function create_table(callback) {
     client.query('CREATE TABLE HIKES (' +
                          'ID VARCHAR(40), ' +
                          'HIKE_DATE DATE, ' +
                          'NAME VARCHAR(40), ' +
                          'DISTANCE VARCHAR(40), ' +
                          'LOCATION VARCHAR(40), ' +
                          'WEATHER VARCHAR(40), ' +
                          'PRIMARY KEY(ID))', callback);
  },
  function insert_default(callback) {
    const hike = {HIKE_DATE: new Date(), NAME: 'Hazard Stevens',
          LOCATION: 'Mt Rainier', DISTANCE: '4,027m vertical', WEATHER:'Bad', ID:
  '12345'};
    client.query('INSERT INTO HIKES set ?', hike, callback);
  }
], function (err, results) {
  if (err) {
    console.log('Exception initializing database.');
    throw err;
  } else {
    console.log('Database initialization complete.');
    init();
  }
});

module.exports = app
```

5. 将以下内容添加到 `routes/hike.js`。这将使路线能够将新的 Hiking 日志插入 HIKES 数据库。

```
const uuid = require('node-uuid');
exports.index = function(req, res) {
  res.app.get('connection').query( 'SELECT * FROM HIKES', function(err,
rows) {
    if (err) {
      res.send(err);
```

```
    } else {
      console.log(JSON.stringify(rows));
      res.render('hike', {title: 'My Hiking Log', hikes: rows});
  }});
};
exports.add_hike = function(req, res){
  const input = req.body.hike;
  const hike = { HIKE_DATE: new Date(), ID: uuid.v4(), NAME: input.NAME,
  LOCATION: input.LOCATION, DISTANCE: input.DISTANCE, WEATHER: input.WEATHER};
  console.log('Request to log hike:' + JSON.stringify(hike));
  req.app.get('connection').query('INSERT INTO HIKES set ?', hike, function(err) {
      if (err) {
        res.send(err);
      } else {
        res.redirect('/hikes');
      }
   });
};
```

6. 将 routes/index.js 的内容替换为以下内容：

```
/*
 * GET home page.
 */

exports.index = function(req, res){
  res.render('index', { title: 'Express' });
};
```

7. 添加以下 jade 模板到 views/hike.jade 中，以提供添加 Hiking 日志的用户界面。

```
extends layout

block content
  h1= title
  p Welcome to #{title}

  form(action="/add_hike", method="post")
    table(border="1")
      tr
        td Your Name
        td
          input(name="hike[NAME]", type="textbox")
```

```
      tr
        td Location
        td
          input(name="hike[LOCATION]", type="textbox")
      tr
        td Distance
        td
          input(name="hike[DISTANCE]", type="textbox")
      tr
        td Weather
        td
          input(name="hike[WEATHER]", type="radio", value="Good")
          | Good
          input(name="hike[WEATHER]", type="radio", value="Bad")
          | Bad
          input(name="hike[WEATHER]", type="radio", value="Seattle", checked)
          | Seattle
      tr
        td(colspan="2")
          input(type="submit", value="Record Hike")

  div
    h3 Hikes
    table(border="1")
      tr
        td Date
        td Name
        td Location
        td Distance
        td Weather
      each hike in hikes
        tr
          td #{hike.HIKE_DATE.toDateString()}
          td #{hike.NAME}
          td #{hike.LOCATION}
          td #{hike.DISTANCE}
          td #{hike.WEATHER}
```

8. 使用 eb deploy 命令部署更改。

```
~/nodejs-example-express-rds$ eb deploy
```

清理

如果使用完 Elastic Beanstalk，则可终止您的环境。

使用 eb terminate 命令终止您的环境以及其中包含的所有资源。

```
~/nodejs-example-express-rds$ eb terminate
The environment "nodejs-example-express-rds-env" and all associated instances will be
 terminated.
To confirm, type the environment name: nodejs-example-express-rds-env
INFO: terminateEnvironment is starting.
...
```

将具有集群功能的 Express 应用程序部署到 Elastic Beanstalk

本教程将引导您使用 Elastic Beanstalk 命令行界面 (EB CLI) 将示例应用程序部署到 Elastic Beanstalk，然后更新应用程序以使用 Express 框架、Amazon 和集群。 ElastiCache集群功能增强了 Web 应用程序的高可用性、性能和安全性。要了解有关亚马逊的更多信息 ElastiCache，请访问 Memcached ElastiCache 的亚马逊是什么？ 在 Amazon f ElastiCache or Memcached 用户指南中。

> **Note**
>
> 此示例创建了 AWS 资源，您可能需要为此付费。有关 AWS 定价的更多信息，请参阅https://aws.amazon.com/pricing/。有些服务属于 AWS 免费使用套餐的一部分。如果您是新客户，则可免费试用这些服务。请参阅https://aws.amazon.com/free/了解更多信息。

先决条件

本教程需要以下先决条件：

- Node.js 运行时
- 默认 Node.js 程序包管理器软件 npm
- Express 命令行生成器
- Elastic Beanstalk 命令行界面 (EB CLI)

有关安装列出的前三个组件和设置本地开发环境的详细信息，请参阅 设置 Node.js 开发环境。在本教程中，您无需安装 AWS 适用于 Node.js 的 SDK，参考主题中也提到了这一点。

有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

创建 Elastic Beanstalk 环境

您的应用程序目录

本教程为应用程序源包使用名为 nodejs-example-express-elasticache 的目录。为本教程创建 nodejs-example-express-elasticache 目录。

```
~$ mkdir nodejs-example-express-elasticache
```

> **Note**
>
> 本章中的每个教程都为应用程序源包使用自己的目录。该目录名称与教程使用的示例应用程序的名称相匹配。

将您当前的工作目录更改为 nodejs-example-express-elasticache。

```
~$ cd nodejs-example-express-elasticache
```

现在，让我们设置运行 Node.js 平台和示例应用程序的 Elastic Beanstalk 环境。我们将使用 Elastic Beanstalk 命令行界面（EB CLI）。

要为您的应用程序配置 EB CLI 存储库，并创建运行 Node.js 平台的 Elastic Beanstalk 环境

1. 使用 eb init 命令创建存储库。

   ```
   ~/nodejs-example-express-elasticache$ eb init --platform node.js --region <region>
   ```

   此命令在名为 .elasticbeanstalk 的文件夹中创建配置文件，该配置文件指定用于为您的应用程序创建环境的设置；并创建以当前文件夹命名的 Elastic Beanstalk 应用程序。
2. 使用 eb create 命令创建运行示例应用程序的环境。

   ```
   ~/nodejs-example-express-elasticache$ eb create --sample nodejs-example-express-elasticache
   ```

   此命令使用 Node.js 平台的默认设置以及以下资源来创建负载均衡环境：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

  > **Note**
  >
  > 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

3. 当环境创建完成后，使用 eb open 命令在默认浏览器中打开环境 URL。

```
~/nodejs-example-express-elasticache$ eb open
```

您现在已经使用示例应用程序创建了 Node.js Elastic Beanstalk 环境。您可以使用自己的应用程序对其进行更新。接下来，我们会更新示例应用程序，以使用 Express 框架。

更新应用程序以使用 Express

更新 Elastic Beanstalk 环境中的示例应用程序以使用 Express 框架。

您可以从 nodejs-example-express-elasticache.zip 下载最终源代码。

更新您的应用程序以使用 Express

在创建具有示例应用程序的环境后，可将其更新为自己的应用程序。在此过程中，首先运行 express 和 npm install 命令，以在您的应用程序目录中设置 Express 框架。

1. 运行 `express` 命令。这将生成 `package.json`、`app.js`，以及几个目录。

   ```
   ~/nodejs-example-express-elasticache$ express
   ```

   在系统提示您是否要继续时，键入 **y**。

   > **Note**
   >
   > 如果 express 命令不起作用，则您可能没有按照前面的先决条件部分所述安装 Express 命令行生成器。或者，可能需要设置本地计算机的目录路径设置才能运行 express 命令。有关设置开发环境的详细步骤，请参阅先决条件部分，以便您可以继续学习本教程。

2. 设置本地依赖项。

   ```
   ~/nodejs-example-express-elasticache$ npm install
   ```

3. （可选）验证 Web 应用程序服务器已启动。

   ```
   ~/nodejs-example-express-elasticache$ npm start
   ```

   您应该可以看到类似于如下所示的输出内容：

   ```
   > nodejs@0.0.0 start /home/local/user/node-express
   > node ./bin/www
   ```

默认情况下，服务器在端口 3000 上运行。要测试，请在另一个终端中运行 `curl http://localhost:3000`，或在本地计算机上打开浏览器并输入 URL 地址 `http://localhost:3000`。

按 Ctrl+C 以停止该服务器。

4. 将 `nodejs-example-express-elasticache/app.js`重命名为 `nodejs-example-express-elasticache/express-app.js`。

```
~/nodejs-example-express-elasticache$ mv app.js express-app.js
```

5. 将 `nodejs-example-express-elasticache/express-app.js` 中的行 `var app = express();` 更新为以下内容：

```
var app = module.exports = express();
```

6. 在本地计算机上，使用以下代码创建一个名为 `nodejs-example-express-elasticache/app.js` 的文件。

```
/**
 * Module dependencies.
 */

 const express = require('express'),
 session = require('express-session'),
 bodyParser = require('body-parser'),
 methodOverride = require('method-override'),
 cookieParser = require('cookie-parser'),
 fs = require('fs'),
 filename = '/var/nodelist',
 app = express();

let MemcachedStore = require('connect-memcached')(session);

function setup(cacheNodes) {
 app.use(bodyParser.raw());
 app.use(methodOverride());
 if (cacheNodes.length > 0) {
   app.use(cookieParser());

   console.log('Using memcached store nodes:');
   console.log(cacheNodes);
```

```
    app.use(session({
      secret: 'your secret here',
      resave: false,
      saveUninitialized: false,
      store: new MemcachedStore({ 'hosts': cacheNodes })
    }));
  } else {
    console.log('Not using memcached store.');
    app.use(session({
      resave: false,
      saveUninitialized: false, secret: 'your secret here'
    }));
  }

  app.get('/', function (req, resp) {
    if (req.session.views) {
      req.session.views++
      resp.setHeader('Content-Type', 'text/html')
      resp.send(`You are session: ${req.session.id}. Views: ${req.session.views}`)
    } else {
      req.session.views = 1
      resp.send(`You are session: ${req.session.id}. No views yet, refresh the page!
`)
    }
  });

  if (!module.parent) {
    console.log('Running express without cluster. Listening on port %d',
  process.env.PORT || 5000)
    app.listen(process.env.PORT || 5000)
  }
}

console.log("Reading elastic cache configuration")
// Load elasticache configuration.
fs.readFile(filename, 'UTF8', function (err, data) {
  if (err) throw err;

  let cacheNodes = []
  if (data) {
    let lines = data.split('\n');
    for (let i = 0; i < lines.length; i++) {
      if (lines[i].length > 0) {
```

```
        cacheNodes.push(lines[i])
      }
    }
 }

 setup(cacheNodes)
});

module.exports = app;
```

7. 将 nodejs-example-express-elasticache/bin/www 文件的内容替换为以下内容：

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

const app = require('../app');
const cluster = require('cluster');
const debug = require('debug')('nodejs-example-express-elasticache:server');
const http = require('http');
const workers = {},
  count = require('os').cpus().length;

function spawn() {
  const worker = cluster.fork();
  workers[worker.pid] = worker;
  return worker;
}


/**
 * Get port from environment and store in Express.
 */

const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

if (cluster.isMaster) {
  for (let i = 0; i < count; i++) {
    spawn();
  }
```

```
  // If a worker dies, log it to the console and start another worker.
  cluster.on('exit', function (worker, code, signal) {
    console.log('Worker ' + worker.process.pid + ' died.');
    cluster.fork();
  });

  // Log when a worker starts listening
  cluster.on('listening', function (worker, address) {
    console.log('Worker started with PID ' + worker.process.pid + '.');
  });

} else {
  /**
   * Create HTTP server.
   */

  let server = http.createServer(app);

  /**
   * Event listener for HTTP server "error" event.
   */

  function onError(error) {
    if (error.syscall !== 'listen') {
      throw error;
    }

    const bind = typeof port === 'string'
      ? 'Pipe ' + port
      : 'Port ' + port;

    // handle specific listen errors with friendly messages
    switch (error.code) {
      case 'EACCES':
        console.error(bind + ' requires elevated privileges');
        process.exit(1);
        break;
      case 'EADDRINUSE':
        console.error(bind + ' is already in use');
        process.exit(1);
        break;
      default:
        throw error;
    }
```

```
  }

  /**
   * Event listener for HTTP server "listening" event.
   */

  function onListening() {
    const addr = server.address();
    const bind = typeof addr === 'string'
      ? 'pipe ' + addr
      : 'port ' + addr.port;
    debug('Listening on ' + bind);
  }

  /**
   * Listen on provided port, on all network interfaces.
   */

  server.listen(port);
  server.on('error', onError);
  server.on('listening', onListening);
}

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  const port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}
```

8. 使用 eb deploy 命令将更改部署到您的 Elastic Beanstalk 环境。

```
~/nodejs-example-express-elasticache$ eb deploy
```

9. 您的环境将在几分钟后进行更新。在环境变为绿色并准备就绪后，刷新 URL 以验证环境是否工作。您应看到一个显示“欢迎使用 Express”的网页。

您可访问运行应用程序的 EC2 实例的日志。有关访问日志的说明，请参阅查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志。

接下来，让我们更新 Express 应用程序以使用亚马逊 ElastiCache。

更新您的 Express 应用程序以使用亚马逊 ElastiCache

1. 在本地计算机上的源包顶级目录中，创建 `.ebextensions` 目录。在此示例中，我们使用的是 `nodejs-example-express-elasticache/.ebextensions`。
2. 使用以下代码段创建配置文件 `nodejs-example-express-elasticache/.ebextensions/elasticache-iam-with-script.config`。有关配置文件的更多信息，请参阅Node.js 配置命名空间。此配置文件会创建一个 IAM 用户（该用户拥有发现 ElastiCache 节点所需的权限），只要缓存发生变化就向某个文件写入数据。您也可以从 nodejs-example-express-elasticache.zip 中复制该文件。有关 ElastiCache属性的更多信息，请参阅示例：ElastiCache。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

```
Resources:
  MyCacheSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: "Lock cache down to webserver access only"
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort:
            Fn::GetOptionSetting:
              OptionName: CachePort
              DefaultValue: 11211
          ToPort:
            Fn::GetOptionSetting:
```

```
            OptionName: CachePort
            DefaultValue: 11211
        SourceSecurityGroupName:
          Ref: AWSEBSecurityGroup
MyElastiCache:
  Type: 'AWS::ElastiCache::CacheCluster'
  Properties:
    CacheNodeType:
      Fn::GetOptionSetting:
        OptionName: CacheNodeType
        DefaultValue: cache.t2.micro
    NumCacheNodes:
      Fn::GetOptionSetting:
        OptionName: NumCacheNodes
        DefaultValue: 1
    Engine:
      Fn::GetOptionSetting:
        OptionName: Engine
        DefaultValue: redis
    VpcSecurityGroupIds:
      -
        Fn::GetAtt:
          - MyCacheSecurityGroup
          - GroupId
AWSEBAutoScalingGroup :
  Metadata :
    ElastiCacheConfig :
      CacheName :
        Ref : MyElastiCache
      CacheSize :
         Fn::GetOptionSetting:
           OptionName : NumCacheNodes
           DefaultValue: 1
WebServerUser :
  Type : AWS::IAM::User
  Properties :
    Path : "/"
    Policies:
      -
        PolicyName: root
        PolicyDocument :
          Statement :
            -
              Effect : Allow
```

```
                Action :
                  - cloudformation:DescribeStackResource
                  - cloudformation:ListStackResources
                  - elasticache:DescribeCacheClusters
                Resource : "*"
  WebServerKeys :
    Type : AWS::IAM::AccessKey
    Properties :
      UserName :
        Ref: WebServerUser

Outputs:
  WebsiteURL:
    Description: sample output only here to show inline string function parsing
    Value: |
      http://`{ "Fn::GetAtt" : [ "AWSEBLoadBalancer", "DNSName" ] }`
  MyElastiCacheName:
    Description: Name of the elasticache
    Value:
      Ref : MyElastiCache
  NumCacheNodes:
    Description: Number of cache nodes in MyElastiCache
    Value:
      Fn::GetOptionSetting:
        OptionName : NumCacheNodes
        DefaultValue: 1

files:
  "/etc/cfn/cfn-credentials" :
    content : |
      AWSAccessKeyId=`{ "Ref" : "WebServerKeys" }`
      AWSSecretKey=`{ "Fn::GetAtt" : ["WebServerKeys", "SecretAccessKey"] }`
    mode : "000400"
    owner : root
    group : root

  "/etc/cfn/get-cache-nodes" :
    content : |
      # Define environment variables for command line tools
      export AWS_ELASTICACHE_HOME="/home/ec2-user/elasticache/$(ls /home/ec2-user/
elasticache/)"
      export AWS_CLOUDFORMATION_HOME=/opt/aws/apitools/cfn
      export PATH=$AWS_CLOUDFORMATION_HOME/bin:$AWS_ELASTICACHE_HOME/bin:$PATH
      export AWS_CREDENTIAL_FILE=/etc/cfn/cfn-credentials
```

```
      export JAVA_HOME=/usr/lib/jvm/jre

      # Grab the Cache node names and configure the PHP page
      aws cloudformation list-stack-resources --stack `{ "Ref" :
 "AWS::StackName" }` --region `{ "Ref" : "AWS::Region" }` --output text | grep
 MyElastiCache | awk '{print $4}' | xargs -I {} aws elasticache describe-cache-
clusters --cache-cluster-id {} --region `{ "Ref" : "AWS::Region" }` --show-
cache-node-info --output text | grep '^ENDPOINT' | awk '{print $2 ":" $3}' >
 `{ "Fn::GetOptionSetting" : { "OptionName" : "NodeListPath", "DefaultValue" : "/
var/www/html/nodelist" } }`
    mode : "000500"
    owner : root
    group : root

  "/etc/cfn/hooks.d/cfn-cache-change.conf" :
    "content": |
      [cfn-cache-size-change]
      triggers=post.update
      path=Resources.AWSEBAutoScalingGroup.Metadata.ElastiCacheConfig
      action=/etc/cfn/get-cache-nodes
      runas=root

sources :
  "/home/ec2-user/elasticache" : "https://elasticache-downloads.s3.amazonaws.com/
AmazonElastiCacheCli-latest.zip"

commands:
  make-elasticache-executable:
    command: chmod -R ugo+x /home/ec2-user/elasticache/*/bin/*

packages :
  "yum" :
    "aws-apitools-cfn"  : []

container_commands:
  initial_cache_nodes:
    command: /etc/cfn/get-cache-nodes
```

3. 在您的本地计算机上，使用以下代码段创建配置文件nodejs-example-express-elasticache/.ebextensions/elasticache_settings.config进行配置 ElastiCache。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
```

```
    CacheNodeType: cache.t2.micro
    NumCacheNodes: 1
    Engine: memcached
    NodeListPath: /var/nodelist
```

4. 在本地计算机上，使用以下代码段替换 nodejs-example-express-elasticache/express-app.js。此文件会从磁盘读取节点列表 (/var/nodelist) 并配置 Express，以便在节点存在的情况下将 memcached 用作会话存储。您的文件应类似以下内容。

```
/**
 * Module dependencies.
 */

var express = require('express'),
    session = require('express-session'),
    bodyParser = require('body-parser'),
    methodOverride = require('method-override'),
    cookieParser = require('cookie-parser'),
    fs = require('fs'),
    filename = '/var/nodelist',
    app = module.exports = express();

var MemcachedStore = require('connect-memcached')(session);

function setup(cacheNodes) {
  app.use(bodyParser.raw());
  app.use(methodOverride());
  if (cacheNodes) {
      app.use(cookieParser());

      console.log('Using memcached store nodes:');
      console.log(cacheNodes);

      app.use(session({
          secret: 'your secret here',
          resave: false,
          saveUninitialized: false,
          store: new MemcachedStore({'hosts': cacheNodes})
      }));
  } else {
    console.log('Not using memcached store.');
    app.use(cookieParser('your secret here'));
    app.use(session());
```

```
  }

  app.get('/', function(req, resp){
  if (req.session.views) {
      req.session.views++
      resp.setHeader('Content-Type', 'text/html')
      resp.write('Views: ' + req.session.views)
      resp.end()
   } else {
      req.session.views = 1
      resp.end('Refresh the page!')
    }
  });

  if (!module.parent) {
      console.log('Running express without cluster.');
      app.listen(process.env.PORT || 5000);
  }
}

// Load elasticache configuration.
fs.readFile(filename, 'UTF8', function(err, data) {
    if (err) throw err;

    var cacheNodes = [];
    if (data) {
        var lines = data.split('\n');
        for (var i = 0 ; i < lines.length ; i++) {
            if (lines[i].length > 0) {
                cacheNodes.push(lines[i]);
            }
        }
    }
    setup(cacheNodes);
});
```

5. 在本地计算机上，使用以下内容更新 package.json：

```
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
```

```
    "morgan": "~1.9.1",
    "connect-memcached": "*",
    "express-session": "*",
    "body-parser": "*",
    "method-override": "*"
  }
```

6. 运行 npm install。

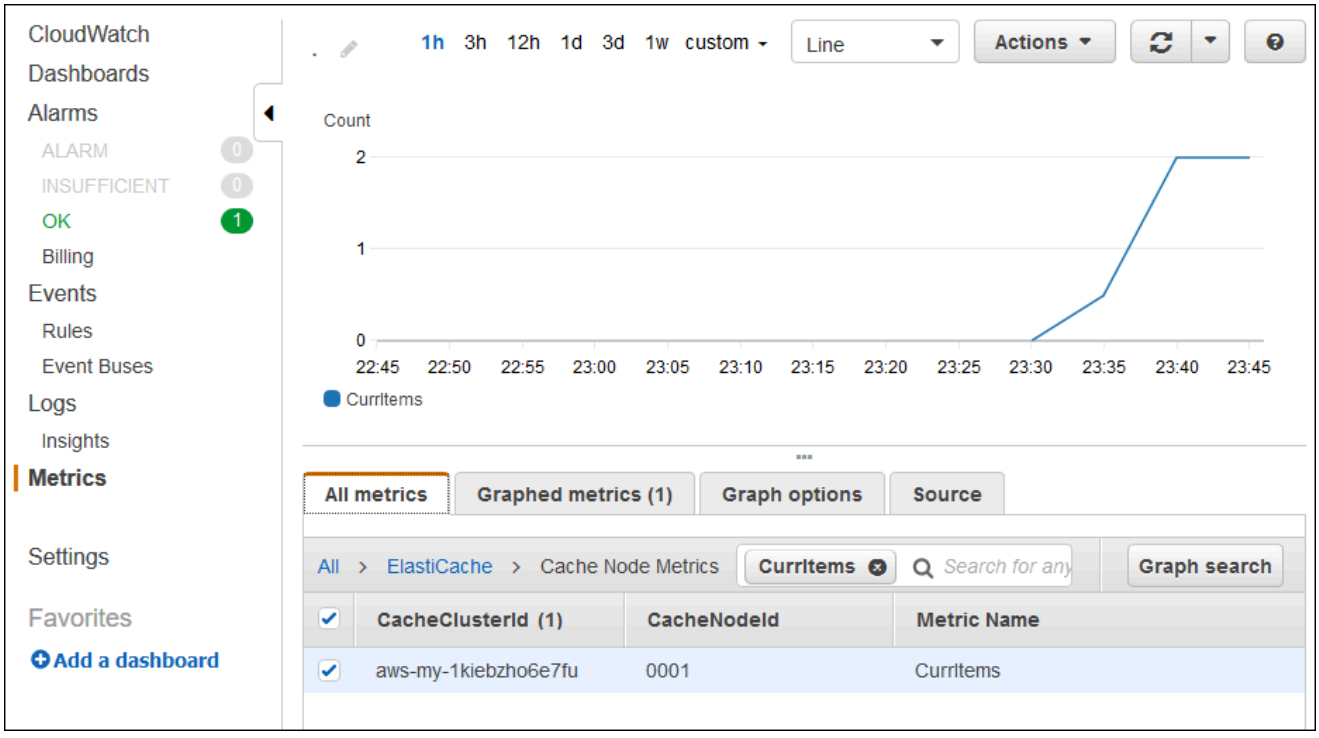
```
~/nodejs-example-express-elasticache$ npm install
```

7. 部署更新的应用程序。

```
~/nodejs-example-express-elasticache$ eb deploy
```

8. 您的环境将在几分钟后进行更新。在环境变为绿色并准备就绪后，验证代码是否正常。

   a. 查看 A mazon CloudWatch 控制台以查看您的 ElastiCache 指标。要查看您的 ElastiCache 指标，请在左侧窗格中选择 “指标”，然后搜索CurrItems。选择 ElastiCache > 缓存节点指标，然后选择您的缓存节点以查看缓存中的项目数量。



> **Note**
>
> 确保您查看的是您的应用程序所部署到的相同地区。

如果您将应用程序 URL 复制并粘贴到其他 Web 浏览器中并刷新页面，则应该会在 5 分钟后看到您的 CurrItem 计数上升。

b. 制作日志的快照。有关检索日志的更多信息，请参阅查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志。

c. 检查日志文件包中的文件 `/var/log/nodejs/nodejs.log`。您应看到类似如下所示的内容：

```
Using memcached store nodes:
[ 'aws-my-1oys9co8zt1uo.1iwtrn.0001.use1.cache.amazonaws.com:11211' ]
```

清理

如果不再希望运行您的应用程序，您可通过终止环境并删除应用程序进行清除。

请使用 `eb terminate` 命令终止环境并使用 `eb delete` 命令删除应用程序。

终止环境

从您在其中创建本地存储库的目录中，运行 `eb terminate`。

```
$ eb terminate
```

此过程可能耗时数分钟。成功终止环境后，Elastic Beanstalk 会立即显示一条消息。

将带 DynamoDB 的 Node.js 应用程序部署到 Elastic Beanstalk

本教程及其示例应用程序 nodejs-example-dynamo.zip 将引导您完成部署 Node.js 应用程序的过程，该应用程序使用 Node.js JavaScript 中的 AWS 软件开发工具包与亚马逊 DynamoDB 服务进行交互。您将创建一个 DynamoDB 表，该表位于与环境分离的数据库或外部数据库中。 AWS Elastic Beanstalk 您还将配置应用程序以使用解耦数据库。在生产环境中，最佳做法是使用与 Elastic Beanstalk 环境解耦的数据库，使其独立于环境的生命周期。这种做法还使您能够执行蓝绿部署。

以下示例应用程序说明下列情况：

- 存储用户提供的文本数据的 DynamoDB 表。
- 用于创建表的配置文件。
- Amazon Simple Notification Service 主题。
- 使用 package.json 文件在部署期间安装程序包。

Sections

先决条件

本教程需要以下先决条件：

- Node.js 运行时
- 默认 Node.js 程序包管理器软件 npm
- Express 命令行生成器
- Elastic Beanstalk 命令行界面 (EB CLI)

有关安装列出的前三个组件和设置本地开发环境的详细信息，请参阅 设置 Node.js 开发环境。在本教程中，您无需安装 AWS 适用于 Node.js 的 SDK，参考主题中也提到了这一点。

有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

创建 Elastic Beanstalk 环境

您的应用程序目录

本教程为应用程序源包使用名为 nodejs-example-dynamo 的目录。为本教程创建 nodejs-example-dynamo 目录。

```
~$ mkdir nodejs-example-dynamo
```

> **Note**
>
> 本章中的每个教程都为应用程序源包使用自己的目录。该目录名称与教程使用的示例应用程序的名称相匹配。

将您当前的工作目录更改为 nodejs-example-dynamo。

```
~$ cd nodejs-example-dynamo
```

现在，让我们设置运行 Node.js 平台和示例应用程序的 Elastic Beanstalk 环境。我们将使用 Elastic Beanstalk 命令行界面（EB CLI）。

要为您的应用程序配置 EB CLI 存储库，并创建运行 Node.js 平台的 Elastic Beanstalk 环境

1. 使用 eb init 命令创建存储库。

   ```
   ~/nodejs-example-dynamo$ eb init --platform node.js --region <region>
   ```

   此命令在名为 .elasticbeanstalk 的文件夹中创建配置文件，该配置文件指定用于为您的应用程序创建环境的设置；并创建以当前文件夹命名的 Elastic Beanstalk 应用程序。
2. 使用 eb create 命令创建运行示例应用程序的环境。

   ```
   ~/nodejs-example-dynamo$ eb create --sample nodejs-example-dynamo
   ```

   此命令使用 Node.js 平台的默认设置以及以下资源来创建负载均衡环境：

   - EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

     各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。

- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

3. 当环境创建完成后，使用 eb open 命令在默认浏览器中打开环境 URL。

```
~/nodejs-example-dynamo$ eb open
```

您现在已经使用示例应用程序创建了 Node.js Elastic Beanstalk 环境。您可以使用自己的应用程序对其进行更新。接下来，我们会更新示例应用程序，以使用 Express 框架。

向环境的实例添加权限

您的应用程序在负载均衡器后方的一个或多个 EC2 实例上运行，为来自 Internet 的 HTTP 请求提供服务。当它收到要求其使用 AWS 服务的请求时，应用程序会使用其运行的实例的权限来访问这些服务。

该示例应用程序使用实例权限向 DynamoDB 表写入数据，并使用 Node.js 中的软件开发工具包向亚马逊 SNS 主题发送通知。 JavaScript 将以下托管式策略添加到默认实例配置文件，以向您环境中的 EC2 实例授予对 DynamoDB 和 Amazon SNS 的访问权限：

- AmazonDynamo数据库 FullAccess
- 亚马逊 SNS FullAccess

向默认实例配置文件添加策略

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择 aws-elasticbeanstalk-ec2 个角色。
3. 在 Permissions（权限）选项卡上，选择 Attach policies（附加策略）。
4. 选择适用于应用程序使用的附加服务的托管策略。在本教程中，请选择 `AmazonSNSFullAccess` 和 `AmazonDynamoDBFullAccess`。
5. 选择附加策略。

请参阅 管理 Elastic Beanstalk 实例配置文件 以了解有关管理实例配置文件的更多信息。

部署示例应用程序

现在，您的环境已准备就绪，可以部署和运行本教程的示例应用程序：nodejs-example-dynamo.zip。

要部署和运行教程示例应用程序

1. 将当前工作目录更改为应用程序目录 `nodejs-example-dynamo`。

   ```
   ~$ cd nodejs-example-dynamo
   ```

2. 将示例应用程序源包 nodejs-example-dynamo.zip 的内容下载并解压缩到应用程序目录中`nodejs-example-dynamo`。
3. 使用 eb deploy 命令将示例应用程序部署到您的 Elastic Beanstalk 环境。

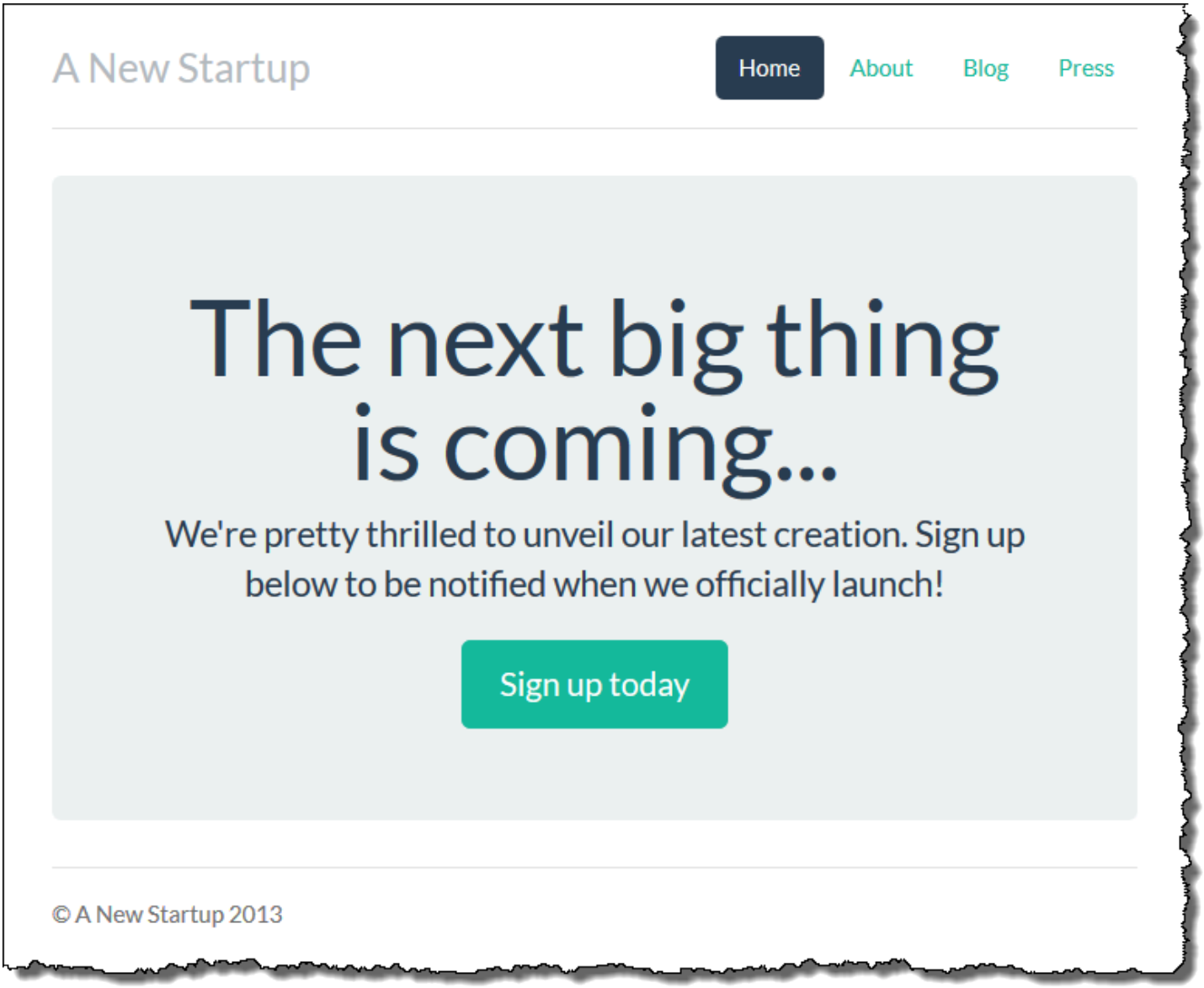
   ```
   ~/nodejs-example-dynamo$ eb deploy
   ```

> **Note**
>
> 默认情况下，该`eb deploy`命令会创建项目文件夹的 ZIP 文件。您可以将 EB CLI 配置为从构建过程部署工件而不是创建项目文件夹 ZIP 文件。有关更多信息，请参阅 部署构件而不是项目文件夹。

4. 当环境创建完成后，使用 eb open 命令在默认浏览器中打开环境 URL。

```
~/nodejs-example-dynamo$ eb open
```

此站点将收集用户联系信息并使用 DynamoDB 表来存储数据。要添加条目，请选择 Sign up today（立即注册），输入名称和电子邮件地址，然后选择 Sign Up!（注册!）。Web 应用程序会将表单内容写入表并触发 Amazon SNS 电子邮件通知。



现在，Amazon SNS 主题已使用通知的占位符电子邮件进行配置。您很快将更新该配置，但同时您可以在 AWS Management Console管理控制台中验证 DynamoDB 表和 Amazon SNS 主题。

查看表

1. 在 DynamoDB 控制台中打开 Tables（表）页面。
2. 找到应用程序创建的表。该名称以 awseb 开头并包含。StartupSignupsTable
3. 选择表，选择 Items（项目），然后选择 Start search（开始搜索）以查看表中所有项目。

在注册站点提交的每个电子邮件地址在表中都有一个条目。除了写入表，应用程序还会向具有两个订阅的 Amazon SNS 主题发送一条消息，一个订阅用于向您发送电子邮件通知，另一个订阅用于某个

Amazon Simple Queue Service 队列，工作线程应用程序可从该队列中读取数据，以处理请求和向感兴趣的客户发送电子邮件。

查看主题

1. 在 Amazon SNS 控制台中打开 Topics（主题）页面。
2. 找到应用程序创建的主题。该名称以 awseb 开头并包含。NewSignupTopic
3. 选择主题以查看其订阅。

应用程序 (`app.js`) 将定义两个路由。根路径 (/) 返回从嵌入式 JavaScript (EJS) 模板呈现的网页，其中包含用户填写的表单以注册其姓名和电子邮件地址。提交表单会将一个 POST 请求和表单数据发送到 `/signup` 路由，后者将一个条目写入到 DynamoDB 表并将一条消息发布到 Amazon SNS 主题，以将注册通知所有者。

示例应用程序包括创建由应用程序使用的 DynamoDB 表、Amazon SNS 主题和 Amazon SQS 队列的配置文件。这可让您创建一个新的环境并立即测试功能，但有将 DynamoDB 表绑定到环境的缺点。对于生产环境，您应在环境外创建 DynamoDB 表，以免在您终止环境或更新其配置时将表丢失。

创建 DynamoDB 表

要将外部 DynamoDB 表用于在 Elastic Beanstalk 中运行的应用程序，请首先在 DynamoDB 中创建表。当您在 Elastic Beanstalk 外部创建表时，它完全独立于 Elastic Beanstalk 和您的 Elastic Beanstalk 环境，并且不会由 Elastic Beanstalk 终止。

使用以下设置创建表：

- Table name（表名称）**`nodejs-tutorial`** –
- Primary key（主键）**`email`** –
- 主键类型 - String

创建 DynamoDB 表

1. 在 DynamoDB 管理控制台中打开 Tables（表）页面。
2. 选择创建表。
3. 键入表名称和主键。
4. 选择主键类型。

5. 选择创建。

更新应用程序的配置文件

更新应用程序源中的配置文件以使用 nodejs-tutorial 表，而不必创建一个新的表。

要更新示例应用程序以供生产用

1. 将当前工作目录更改为应用程序目录 `nodejs-example-dynamo`。

```
~$ cd nodejs-example-dynamo
```

2. 打开 `.ebextensions/options.config` 并更改以下设置的值：

   - NewSignupEmail— 您的电子邮件地址。
   - STARTUP_SIGNUP_TABLE – nodejs-tutorial

   Example .ebextensions/options.config

```
option_settings:
  aws:elasticbeanstalk:customoption:
    NewSignupEmail: you@example.com
  aws:elasticbeanstalk:application:environment:
    THEME: "flatly"
    AWS_REGION: '`{"Ref" : "AWS::Region"}`'
    STARTUP_SIGNUP_TABLE: nodejs-tutorial
    NEW_SIGNUP_TOPIC: '`{"Ref" : "NewSignupTopic"}`'
  aws:elasticbeanstalk:container:nodejs:
    ProxyServer: nginx
  aws:elasticbeanstalk:container:nodejs:staticfiles:
    /static: /static
  aws:autoscaling:asg:
    Cooldown: "120"
  aws:autoscaling:trigger:
    Unit: "Percent"
    Period: "1"
    BreachDuration: "2"
    UpperThreshold: "75"
    LowerThreshold: "30"
    MeasureName: "CPUUtilization"
```

这将为应用程序应用以下配置：

- Amazon SNS 主题用于通知的电子邮件地址已设置为您的地址，或您在 `options.config` 文件中输入的地址。
- 将使用 nodejs-tutorial 表，而不是由 `.ebextensions/create-dynamodb-table.config` 创建的表。

3. 删除 `.ebextensions/create-dynamodb-table.config`。

```
~/nodejs-tutorial$ rm .ebextensions/create-dynamodb-table.config
```

您下一次部署应用程序时，此配置文件创建的表将被删除。

4. 使用 eb deploy 命令将更新的应用程序部署到您的 Elastic Beanstalk 环境。

```
~/nodejs-example-dynamo$ eb deploy
```

5. 当环境创建完成后，使用 eb open 命令在默认浏览器中打开环境 URL。

```
~/nodejs-example-dynamo$ eb open
```

部署时，Elastic Beanstalk 将更新 Amazon SNS 主题的配置并删除它在您部署应用程序的第一个版本时创建的 DynamoDB 表。

现在，当您终止环境时，nodejs-tutorial 表将不会被删除。这可让您执行蓝/绿部署、修改配置文件或关闭您的网站而不会有丢失数据的风险。

在浏览器中打开您的站点并验证表单是否按预期运行。创建一些条目，然后检查 DynamoDB 控制台以验证表。

查看表

1. 在 DynamoDB 控制台中打开 Tables（表）页面。
2. 查找 nodejs-tutorial 表。
3. 选择表，选择 Items（项目），然后选择 Start search（开始搜索）以查看表中所有项目。

您也可以看到 Elastic Beanstalk 删除了之前创建的表。

为您的环境配置高可用性

最后，使用较高的最低实例计数配置您环境的 Auto Scaling 组。请始终至少运行两个实例，以防止您环境中的 Web 服务器发生单点故障，并支持您在不中断站点服务的情况下部署更改。

配置您的环境的 Auto Scaling 组以获得高可用性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在 Auto Scaling group（Auto Scaling 组）部分中，将 Min instances（最小实例数）设置为 **2**。
6. 要保存更改，请选择页面底部的 Apply（应用）。

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

还可以删除由您创建的外部 DynamoDB 表。

删除 DynamoDB 表

1. 在 DynamoDB 控制台中打开 Tables（表）页面。
2. 选择表。
3. 选择操作，然后选择删除表。
4. 选择 Delete（删除）。

后续步骤

示例应用程序使用配置文件来配置软件设置并创建 AWS 资源作为环境的一部分。请参阅 使用配置文件 (.ebextensions) 进行高级环境自定义 以了解有关配置文件及其用途的详细信息。

本教程的示例应用程序使用适用于 Node.js 的 Express Web 框架。有关 Express 的更多信息，请参阅 expressjs.com 上的官方文档。

最后，如果您计划在生产环境中使用应用程序，请为环境配置自定义域名并为安全连接启用 HTTPS。

向 Node.js 应用程序环境中添加 Amazon RDS 数据库实例

您可以使用 Amazon Relational Database Service（Amazon RDS）数据库实例来存储由应用程序收集和修改的数据。数据库可以耦合到您的环境并由 Elastic Beanstalk 进行管理，也可以被创建为解耦数据库并由另一项服务进行外部管理。本主题提供使用 Elastic Beanstalk 控制台创建 Amazon RDS 的说明。数据库将耦合到您的环境并由 Elastic Beanstalk 进行管理。有关将 Amazon RDS 与 Elastic Beanstalk 集成的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

小节目录

向环境中添加数据库实例

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

添加一个数据库实例大约需要 10 分钟。环境更新完成后，您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息：

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

有关与 Elastic Beanstalk 环境耦合的数据库实例配置的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

下载驱动程序

在项目的 package.json 文件中的 dependencies 下面添加数据库驱动程序。

Example **package.json** - Express with MySQL

```
{
  "name": "my-app",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
    "ejs": "latest",
    "aws-sdk": "latest",
    "express": "latest",
    "body-parser": "latest",
    "mysql": "latest"
  },
  "scripts": {
    "start": "node app.js"
  }
}
```

Node.js 的常见驱动程序包

- MySQL – mysql
- PostgreSQL – node-postgres
- SQL Server — node-mssql
- Oracle – node-oracledb

连接到数据库

Elastic Beanstalk 在环境属性中提供所连数据库实例的连接信息。使用 process.env.*VARIABLE* 可读取这些属性并配置数据库连接。

Example app.js – MySQL 数据库连接

```
var mysql = require('mysql');
```

```
var connection = mysql.createConnection({
  host     : process.env.RDS_HOSTNAME,
  user     : process.env.RDS_USERNAME,
  password : process.env.RDS_PASSWORD,
  port     : process.env.RDS_PORT
});

connection.connect(function(err) {
  if (err) {
    console.error('Database connection failed: ' + err.stack);
    return;
  }

  console.log('Connected to database.');
});

connection.end();
```

有关使用 node-mysql 构造连接字符串的更多信息，请参阅 npmjs.org/package/mysql。

Resources（资源）

在开发 Node.js 应用程序时，您可以在多个地方获得额外的帮助：

资源	描述
GitHub。	使用 GitHub 安装适用于 Node.js 的AWS开发工具包。
Node.js 开发论坛	发布您的问题并获取反馈。
AWS SDK for Node.js（开发人员预览）	示例代码、文档、工具和其他资源的一站式商店。

在 Elastic Beanstalk 上创建和部署 PHP 应用程序

AWS Elastic Beanstalk for PHP 让您可以使用 Amazon Web Services 轻松部署、管理和扩展您的 PHP Web 应用程序。本章提供将 PHP Web 应用程序部署到 Elastic Beanstalk 的说明。您可以使用 Elastic Beanstalk 命令行界面 (EB CLI) 或使用 Elastic Beanstalk 控制台在短短几分钟内部署应用程序。

本章提供以下教程：

- QuickStart 对于 PHP— 使用 EB CLI 部署 H ello World PHP 应用程序。
- 示例应用程序和教程— CakePHP 和 Symfony 等常见框架的深入教程，以及向你的 PHP 应用程序环境中添加 Amazon RDS 实例。

如果您需要 PHP 应用程序开发的帮助，则可以访问以下几个资源：

- GitHub— 使用安装适用于 PHP 的 AWS SDK GitHub。
- PHP 开发人员中心 — 获取示例代码、文档、工具和其他资源的一站式商店。
- AWS SDK for PHP 常见问题解答 — 获取常见问题的答案。

主题

- QuickStart: 将 PHP 应用程序部署到 Elastic Beanstalk
- 设置 PHP 开发环境
- 使用 Elastic Beanstalk PHP 平台
- 更多 PHP 示例应用程序和教程

QuickStart: 将 PHP 应用程序部署到 Elastic Beanstalk

本 QuickStart 教程将引导您完成创建 PHP 应用程序并将其部署到 AWS Elastic Beanstalk 环境中的过程。

> **Note**
>
> 本 QuickStart 教程仅用于演示目的。请勿将本教程中创建的应用程序用于生产流量。

Sections

- 你的 AWS 账户
- 先决条件
- 步骤 1：创建 PHP 应用程序
- 步骤 2：在本地运行应用程序

- 步骤 3：使用 EB CLI 部署您的 PHP 应用程序
- 第 4 步：在 Elastic Beanstalk 上运行你的应用程序
- 第 5 步：清理
- AWS 您的应用程序的资源
- 后续步骤
- 使用 Elastic Beanstalk 控制台进行部署

你的 AWS 账户

如果您还不是 AWS 客户，则需要创建一个 AWS 帐户。注册后，您就可以访问 Elastic Beanstalk AWS 和其他所需的服务。

如果您已经有一个 AWS 帐户，则可以继续前进先决条件 。

创建 AWS 账户

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

先决条件

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

EB CLI

本教程使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅 安装 EB CLI 和 配置 EB CLI。

PHP

按照 PHP 网站上的安装和配置操作，在本地计算机上安装 PHP。

步骤 1：创建 PHP 应用程序

在这个例子中，我们创建了一个 H ello World PHP 应用程序。可以以最小的开销创建 PHP 应用程序。

创建项目目录。

```
~$ mkdir eb-php
~$ cd eb-php
```

接下来，在项目目录中创建一个`index.php`文件。运行 PHP 时默认提供此文件。

```
~/eb-php/
|-- index.php
```

将以下内容添加到您的`index.php`文件中。

Example **`~/eb-php/index.php`**

```
echo "Hello Elastic Beanstalk! This is a PHP application.";
```

步骤 2：在本地运行应用程序

运行以下命令以在本地运行应用程序。

```
~/eb-php$ php -S localhost:5000
```

`http://localhost:5000`在您的网络浏览器中输入 URL 地址。浏览器应显示 “你好 Elastic Beanstalk！这是一个 PHP 应用程序。”

步骤 3：使用 EB CLI 部署您的 PHP 应用程序

运行以下命令为此应用程序创建 Elastic Beanstalk 环境。

创建环境并部署 PHP 应用程序

1. 使用 eb init 命令，初始化 EB CLI 存储库。

   ```
   ~/eb-php$ eb init -p php php-tutorial --region us-east-2
   ```

   此命令将创建一个名为的应用程序，`php-tutorial`并将您的本地存储库配置为使用最新 PHP 平台版本创建环境。

2. （可选）再次运行 eb init 以配置默认密钥对，以便使用 SSH 连接到运行您的应用程序的 EC2 实例。

   ```
   ~/eb-php$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

   如果您已有密钥对，请选择一个，或按提示创建一个。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。

3. 创建环境并使用 eb create 将应用程序部署到此环境中。Elastic Beanstalk 会自动为您的应用程序生成一个 zip 文件，并将其部署到环境中的 EC2 实例。部署应用程序后，Elastic Beanstalk 在端口 5000 上启动它。

   ```
   ~/eb-php$ eb create php-env
   ```

Elastic Beanstalk 创建您的环境大约需要五分钟。

第 4 步：在 Elastic Beanstalk 上运行你的应用程序

创建环境的过程完成后，使用打开您的网站eb open。

```
~/eb-php$ eb open
```

恭喜您！你已经使用 Elastic Beanstalk 部署了一个 PHP 应用程序！这将使用为应用程序创建的域名打开一个浏览器窗口。

第 5 步：清理

完成应用程序的使用后，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源。

要使用 EB CLI 终止您的 Elastic Beanstalk 环境，请运行以下命令。

```
~/eb-php$ eb terminate
```

AWS 您的应用程序的资源

您刚刚创建了一个单实例应用程序。它可用作带有单个 EC2 实例的简单示例应用程序，因此不需要负载平衡或 auto Scaling。对于单实例应用程序，Elastic Beanstalk 会创建以下资源： AWS

- EC2 实例 - 配置来在您选择的平台上运行 Web 应用程序的 Amazon EC2 虚拟机。

  各平台运行一组不同的软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 nginx 作为在 Web 应用程序前处理 Web 流量的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的传入流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。

- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

Elastic Beanstalk 管理所有这些资源。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

后续步骤

有了运行应用程序的环境以后，您随时可以部署新的应用程序版本或不同的应用程序。部署新应用程序版本非常快，因为不需要配置或重新启动 EC2 实例。您还可以使用 Elastic Beanstalk 控制台探索您的新环境。有关详细步骤，请参阅本指南入门一章中的探索您的环境。

> **尝试更多教程**
>
> 如果您想尝试其他包含不同示例应用程序的教程，请参阅更多 PHP 示例应用程序和教程。

部署一两个示例应用程序并准备好开始在本地开发和运行 PHP 应用程序之后，请参阅设置 PHP 开发环境。

使用 Elastic Beanstalk 控制台进行部署

您也可以使用 Elastic Beanstalk 控制台启动示例应用程序。有关详细步骤，请参阅本指南入门一章中的创建示例应用程序。

设置 PHP 开发环境

设置 PHP 开发环境以在本地测试应用程序，然后再将它部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

有关适用于所有语言的常见设置步骤和工具，请参阅配置您的开发计算机。

小节目录

- 安装 IDE 或文本编辑器

安装 PHP

安装 PHP 和一些常用扩展。如果您没有特别的要求，请获取最新版本。根据平台和可用的程序包管理器，步骤可能有所不同。

在 Amazon Linux 上，请使用 yum：

```
$ sudo yum install php
$ sudo yum install php-mbstring
$ sudo yum install php-intl
```

> **Note**
>
> 要获取与 Elastic Beanstalk PHP 平台版本上的版本匹配的特定 PHP 软件包版本，请使用命令 `yum search php` 查找可用软件包版本，例如 `php72`、`php72-mbstring` 和 `php72-intl`。然后，使用 `sudo yum install` *`package`* 安装它们。

在 Ubuntu 上，使用 apt：

```
$ sudo apt install php-all-dev
$ sudo apt install php-intl
$ sudo apt install php-mbstring
```

在 OS X 上，请使用 brew：

```
$ brew install php
$ brew install php-intl
```

> **Note**
>
> 要获取与 Elastic Beanstalk PHP 平台版本上的版本匹配的特定 PHP 软件包版本，请参阅可用 PHP 版本的 Homebrew Formulae，例如 `php@7.2`。然后，使用 `brew install` *`package`* 安装它们。
>
> 根据版本，`php-intl` 可能包含在主 PHP 软件包中，不作为单独的软件包存在。

在 Windows 10 上，安装 Windows Subsystem for Linux 以获取 Ubuntu 并使用 apt 安装 PHP。对于早期版本，请访问下载页面 (windows.php.net) 以获取 PHP，并阅读此页面以了解有关扩展的信息。

安装 PHP 后，请重新打开终端并运行 `php --version`，以确保已安装新版本并为默认值。

安装 Composer

Composer 是用于 PHP 的依赖项管理器。您可以使用它来安装库、跟踪应用程序的依赖项并为热门 PHP 框架生成项目。

使用来自 getcomposer.org 的 PHP 脚本安装 Composer。

```
$ curl -s https://getcomposer.org/installer | php
```

安装程序将在当前目录中生成 PHAR 文件。将此文件移动到环境 PATH 中的位置以便将此文件用作可执行文件。

```
$ mv composer.phar ~/.local/bin/composer
```

使用 `require` 命令安装库。

```
$ composer require twig/twig
```

Composer 会将您在本地安装的库添加到您的项目的`composer.json` 文件。在部署项目代码时，Elastic Beanstalk 将使用 Composer 在您的环境中的应用程序实例上安装此文件中列出的库。

如果您在安装 Composer 时遇到问题，请参阅 Composer 文档。

安装 AWS SDK for PHP

如果您需要在应用程序内管理 AWS 资源，请安装 AWS SDK for PHP。例如，借助 SDK for PHP，您可以使用 Amazon DynamoDB（DynamoDB）来存储用户和会话信息，而无需创建关系数据库。

使用 Composer 安装 SDK for PHP。

```
$ composer require aws/aws-sdk-php
```

请访问 AWS SDK for PHP 主页以了解更多信息和安装说明。

安装 IDE 或文本编辑器

集成开发环境（IDE）提供了便于应用程序开发的大量功能。如果您还没使用 IDE 进行过 PHP 开发，请尝试 Eclipse 和 PHPStorm，看哪个更适合您。

- 安装 Eclipse
- 安装 PhpStorm

> **Note**
>
> IDE 可以将您可能不希望提交到源代码控制的文件添加到项目文件夹中。要防止将这些文件提交到源代码控制，请使用 `.gitignore` 或您的源代码控制工具的同类功能。

如果您只是希望开始编码而不需要所有 IDE 功能，请考虑安装 Sublime Text。

使用 Elastic Beanstalk PHP 平台

AWS Elastic Beanstalk 支持适用于不同版本的 PHP 编程语言的多种平台。这些平台支持可以单独运行或在 Composer 下运行的 PHP Web 应用程序。在 AWS Elastic Beanstalk 平台文档中的 PHP 中了解详情。

Elastic Beanstalk 提供了配置选项，可供您用于自定义在 Elastic Beanstalk 环境中的 EC2 实例上运行的软件。您可配置应用程序所需的环境变量，启用到 Amazon S3 的日志轮换，将应用程序源中包含静态文件的文件夹映射至代理服务器所提供的路径，并设置常见 PHP 初始化设置。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

如果使用了 Composer，您可以将 `composer.json` 文件包含在源包中以在部署期间安装程序包。

对于未提供为配置选项的高级 PHP 配置和 PHP 设置，您可以使用配置文件来提供 `INI` 文件，该文件可扩展并覆盖由 Elastic Beanstalk 应用的默认设置或者安装其他扩展。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅 the section called “扩展 Linux 平台”。

Amazon Linux 2 上的 PHP 8.1 注意事项

如果使用的是 Amazon Linux 2 上的 PHP 8.1 平台分支，请阅读本节。

Amazon Linux 2 上的 PHP 8.1 注意事项

> **Note**
>
> 本主题中的信息仅适用于 Amazon Linux 2 上的 PHP 8.1 平台分支。它不适用于基于 AL2023 的 PHP 平台分支。它也不适用于 Amazon Linux 2 上的 PHP 8.0 平台分支。

Elastic Beanstalk 将 EC2 实例上与 Amazon Linux 2 上的 PHP 8.1 平台分支相关的 RPM 程序包存储在本地目录中，而不是在 Amazon Linux 存储库中。您可以使用 rpm -i 来安装程序包。从 PHP 8.1 平台版本 3.5.0 开始，Elastic Beanstalk 会将与 PHP 8.1 相关的 RPM 程序包存储在以下本地 EC2 目录中。

`/opt/elasticbeanstalk/RPMS`

下面的示例将安装 php-debuginfo 程序包。

```
$rpm -i /opt/elasticbeanstalk/RPMS/php-debuginfo-8.1.8-1.amzn2.x86_64.rpm
```

程序包名称中的版本将根据 EC2 本地目录 `/opt/elasticbeanstalk/RPMS` 中列出的实际版本而有所不同。使用相同的语法安装其他 PHP 8.1 RPM 程序包。

展开以下部分以显示我们提供的 RPM 程序包列表。

RPM 程序包

以下列表提供了 Elastic Beanstalk PHP 8.1 平台在 Amazon Linux 2 上提供的 RMP 程序包。它们位于本地目录 `/opt/elasticbeanstalk/RPMS` 下。

列出的程序包名称中的版本号 8.1.8-1 和 3.7.0-1 只是举例说明。

- `php-8.1.8-1.amzn2.x86_64.rpm`
- `php-bcmath-8.1.8-1.amzn2.x86_64.rpm`

- php-cli-8.1.8-1.amzn2.x86_64.rpm
- php-common-8.1.8-1.amzn2.x86_64.rpm
- php-dba-8.1.8-1.amzn2.x86_64.rpm
- php-dbg-8.1.8-1.amzn2.x86_64.rpm
- php-debuginfo-8.1.8-1.amzn2.x86_64.rpm
- php-devel-8.1.8-1.amzn2.x86_64.rpm
- php-embedded-8.1.8-1.amzn2.x86_64.rpm
- php-enchant-8.1.8-1.amzn2.x86_64.rpm
- php-fpm-8.1.8-1.amzn2.x86_64.rpm
- php-gd-8.1.8-1.amzn2.x86_64.rpm
- php-gmp-8.1.8-1.amzn2.x86_64.rpm
- php-intl-8.1.8-1.amzn2.x86_64.rpm
- php-ldap-8.1.8-1.amzn2.x86_64.rpm
- php-mbstring-8.1.8-1.amzn2.x86_64.rpm
- php-mysqlnd-8.1.8-1.amzn2.x86_64.rpm
- php-odbc-8.1.8-1.amzn2.x86_64.rpm
- php-opcache-8.1.8-1.amzn2.x86_64.rpm
- php-pdo-8.1.8-1.amzn2.x86_64.rpm
- php-pear-1.10.13-1.amzn2.noarch.rpm
- php-pgsql-8.1.8-1.amzn2.x86_64.rpm
- php-process-8.1.8-1.amzn2.x86_64.rpm
- php-pspell-8.1.8-1.amzn2.x86_64.rpm
- php-snmp-8.1.8-1.amzn2.x86_64.rpm
- php-soap-8.1.8-1.amzn2.x86_64.rpm
- php-sodium-8.1.8-1.amzn2.x86_64.rpm
- php-xml-8.1.8-1.amzn2.x86_64.rpm
- php-pecl-imagick-3.7.0-1.amzn2.x86_64.rpm
- php-pecl-imagick-debuginfo-3.7.0-1.amzn2.x86_64.rpm
- php-pecl-imagick-devel-3.7.0-1.amzn2.noarch.rpm

您可以使用 PEAR 和 PECL 程序包来安装常用扩展。有关 PEAR 的更多信息，请参阅 PEAR PHP Extension and Application Repository 网站。有关 PECL 的更多信息，请参阅 PECL 扩展网站。

以下示例命令将安装 Memcached 扩展。

```
$pecl install memcache
```

或者您也可以使用以下方法：

```
$pear install pecl/memcache
```

以下示例命令将安装 Redis 扩展。

```
$pecl install redis
```

或者您也可以使用以下方法：

```
$pear install pecl/redis
```

配置 PHP 环境

您可以使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，配置应用程序可以从环境中读取的变量以及更改 PHP 设置。

在 Elastic Beanstalk 控制台中配置 PHP 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

PHP 设置

- Proxy server（代理服务器）– 要在环境实例上使用的代理服务器。默认情况下，使用 nginx。

- Document root（文档根目录）– 包含您站点的默认页面的文件夹。如果您的欢迎页面不位于源包的根目录，请指定包含该页面且与根路径相关的文件夹。例如，如果欢迎页面位于名为 `/public` 的文件夹中，则为 `public`。
- Memory limit（内存限制）– 允许脚本分配的最大内存量。例如，`512M`。
- Zlib output compression（Zlib 输出压缩）– 设置为 `On` 可压缩响应。
- Allow URL fopen（允许 URL fopen）– 设置为 `Off` 可防止脚本从远程位置下载文件。
- Display errors（显示错误）– 设置为 `On` 可显示要调试的内部错误消息。
- Max execution time（最长执行时间）– 脚本在被环境终止前允许运行的最长时间，单位为秒。

日志选项

“日志选项”部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

环境属性

在环境属性部分，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。这些设置会以密钥值对的方式传递到应用程序。

您的应用程序代码可使用 `$_SERVER` 或 `get_cfg_var` 函数来访问环境属性。

```
$endpoint = $_SERVER['API_ENDPOINT'];
```

请参阅环境属性和其他软件设置了解更多信息。

aws:elasticbeanstalk:container:php:phpini 命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间中。

您可以使用 `aws:elasticbeanstalk:environment:proxy` 命名空间以选择环境的代理服务器。

您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间来配置环境代理以提供静态文件。您可以定义虚拟路径到应用程序目录的映射。

PHP 平台在 `aws:elasticbeanstalk:container:php:phpini` 命名空间中定义一些选项，包括在 Elastic Beanstalk 控制台中未提供的选项。`composer_options` 设置在通过 `composer.phar install` 使用 Composer 安装依赖项时使用的自定义选项。有关包括可用选项在内的更多信息，请转到 http://getcomposer.org/doc/03-cli.md#install。

以下示例配置文件指定一个静态文件选项，该选项将名为 `staticimages` 的目录映射到路径 `/images`，并显示 `aws:elasticbeanstalk:container:php:phpini` 命名空间中提供的每个选项的设置：

Example .ebextensions/php-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /images: staticimages
  aws:elasticbeanstalk:container:php:phpini:
    document_root: /public
    memory_limit: 128M
    zlib.output_compression: "Off"
    allow_url_fopen: "On"
    display_errors: "Off"
    max_execution_time: 60
    composer_options: vendor/package
```

> **Note**
>
> `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间未在 Amazon Linux AMI PHP 平台分支（在 Amazon Linux 2 之前）上定义。

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI来配置选项。请参阅配置选项了解更多信息。

安装您的应用程序的依赖项

您的应用程序可能具有对其他 PHP 包的依赖项。您可以配置应用程序以在环境的 Amazon Elastic Compute Cloud（Amazon EC2）实例上安装这些依赖项。或者，您也可以将应用程序的依赖项包含在源包中并将它们与应用程序一起部署。以下部分讨论这两种方法。

使用 Composer 文件在实例上安装依赖项

借助位于项目源的根目录中的 `composer.json` 文件，可以使用 Composer 在环境的 Amazon EC2 实例上安装应用程序所需的包。

Example composer.json

```
{
    "require": {
        "monolog/monolog": "1.0.*"
    }
}
```

如果 `composer.json` 文件存在，Elastic Beanstalk 将运行 `composer.phar install` 以安装依赖项。您可以通过在 `aws:elasticbeanstalk:container:php:phpini` 命名空间中设置 `composer_options` 选项来添加要追加到命令的选项。

使依赖项包含在源包中

如果您的应用程序具有大量依赖项，则安装它们可能需要很长时间。这可能会增加部署和扩展操作，因为需要在每个新实例上安装依赖项。

要避免对部署时间带来负面影响，请在您的开发环境中使用 Composer 解析依赖项并将其安装到 `vendor` 文件夹中。

使依赖项包含在您的应用程序源包中

1. 运行以下命令：

   ```
   % composer install
   ```

2. 使生成的 `vendor` 文件夹包含在您的应用程序源包的根目录中。

当 Elastic Beanstalk 在实例上找到 vendor 文件夹时，它会忽略 composer.json 文件（如果存在）。然后，您的应用程序将使用 vendor 文件夹中的依赖项。

更新 Composer

如果您在尝试安装包含 Composer 文件的软件包时看到错误，或者无法使用最新的平台版本，则可能需要更新 Composer。在平台更新之间，您可以使用.ebextensions文件夹中的配置文件在环境实例中更新 Composer。

您可以使用以下配置自行更新 Composer。

```
commands:
  01updateComposer:
    command: /usr/bin/composer.phar self-update 2.7.0
```

以下选项设置设置COMPOSER_HOME环境变量，该变量用于配置 Composer 缓存的位置。

```
option_settings:
  - namespace: aws:elasticbeanstalk:application:environment
    option_name: COMPOSER_HOME
    value: /home/webapp/composer-home
```

您可以将两者合并到.ebextensions文件夹中的同一个配置文件中。

Example .ebextensions/composer.config

```
commands:
  01updateComposer:
    command: /usr/bin/composer.phar self-update 2.7.0

option_settings:
  - namespace: aws:elasticbeanstalk:application:environment
    option_name: COMPOSER_HOME
    value: /home/webapp/composer-home
```

> **Note**
>
> 由于 2024 年 2 月 22 日 AL2023 平台版本和 2024 年 2 月 28 日的 AL2 平台版本对 Composer 安装进行了更新，因此，如果COMPOSER_HOME在执行自我更新时设置，Composer 自我更新可能会失败。

以下组合命令将无法执行：`export COMPOSER_HOME=/home/webapp/composer-home && /usr/bin/composer.phar self-update 2.7.0`
但是，前面的示例将起作用。在前面的示例中，的选项设置`COMPOSER_HOME`不会传递给`01updateComposer`执行，也不会在执行自我更新命令时进行设置。

Important

如果您在 `composer.phar self-update` 命令中省略了版本号，则在您每次部署到源代码时，以及 Auto Scaling 预配置新实例时，Composer 将更新到可用的最新版本。如果发行了与您应用程序不兼容的 Composer 版本，这可能导致扩展操作和部署失败。

有关 Elastic Beanstalk PHP 平台，包括 Composer 的版本的更多信息，请参阅文档 AWS Elastic Beanstalk 平台中的 PHP 平台版本。

扩展 php.ini

使用带有 `files` 块的配置文件可将 `.ini` 文件添加到您的环境中的实例上的 `/etc/php.d/` 中。主配置文件 `php.ini` 按字母顺序从此文件夹中的文件拉入设置。默认情况下，此文件夹中的文件将启用大量扩展。

Example .ebextensions/mongo.config

```
files:
  "/etc/php.d/99mongo.ini":
    mode: "000755"
    owner: root
    group: root
    content: |
      extension=mongo.so
```

更多 PHP 示例应用程序和教程

要开始使用 PHP 应用程序 AWS Elastic Beanstalk，您只需要一个应用程序源包，将其作为第一个应用程序版本上传并部署到环境中。本QuickStart 对于 PHP主题将引导您使用 EB CLI 启动示例 PHP 应用程序。本节提供更深入的教程。

PHP 教程

- 将 Laravel 应用程序部署到 Elastic Beanstalk
- 将 CakePHP 应用程序部署到 Elastic Beanstalk
- 将 Symfony 应用程序部署到 Elastic Beanstalk
- 将带有外部 Amazon RDS 数据库的高可用性 PHP 应用程序部署到 Elastic Beanstalk
- 将带有外部 Amazon RDS 数据库的高可用性 WordPress 网站部署到 Elastic Beanstalk
- 将带有外部 Amazon RDS 数据库的高可用性 Drupal 网站部署到 Elastic Beanstalk
- 向 PHP 应用程序环境中添加 Amazon RDS 数据库实例

将 Laravel 应用程序部署到 Elastic Beanstalk

Laravel 是一个适用于 PHP 的开源 model-view-controller （MVC）框架。本教程将引导您完成生成 Laravel 应用程序、将其部署到 AWS Elastic Beanstalk 环境以及将其配置为连接到亚马逊关系数据库服务 (Amazon RDS) 数据库实例的过程。

Sections

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

Laravel 6 需要 PHP 7.2 或更高版本。它还需要 Laravel 官方文档中的服务器要求主题中列出的 PHP 扩展。按照设置 PHP 开发环境主题中的说明安装 PHP 和 Composer。

如需 Laravel 支持和维护信息，请参阅 Laravel 官方文档中的支持策略主题。

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 PHP 平台并接受默认设置和示例代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。

- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

安装 Laravel 并生成网站

Composer 可使用一个命令安装 Laravel 并创建工作项目：

```
~$ composer create-project --prefer-dist laravel/laravel eb-laravel
```

Composer 安装 Laravel 及其依赖项，并生成默认项目。

如果在安装 Laravel 时遇到任何问题，请参阅官方文档中的安装主题：https://laravel.com/docs/6.x

部署您的应用程序

创建包含由 Composer 创建的文件的源包。以下命令将创建名为 `laravel-default.zip` 的源包。它将排除 `vendor` 文件夹中的文件，因为这些文件会占用大量空间并且对于将您的应用程序部署到 Elastic Beanstalk 不是必需的。

```
~/eb-laravel$ zip ../laravel-default.zip -r * .[^.]* -x "vendor/*"
```

将源包上传到 Elastic Beanstalk 以将 Laravel 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在“区域”列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

> **Note**
>
> 为了进一步优化源包，请初始化一个 Git 存储库并使用 `git archive` 命令创建源包。默认 Laravel 项目包含一个 `.gitignore` 文件，该文件指示 Git 排除 `vendor` 文件夹以及部署时不需要的其他文件。

配置 Composer 设置

部署完成后，单击 URL 以在浏览器中打开 Laravel 应用程序：

这是什么? 默认情况下，Elastic Beanstalk 提供项目在网站根路径下的根目录。但在这种情况下，默认页面 (`index.php`) 位于 `public` 文件夹的下一级。您可通过将 `/public` 添加至 URL 加以验证。例如，`http://laravel.us-east-2.elasticbeanstalk.com/public`。

要在根路径中提供 Laravel 应用程序，请使用 Elastic Beanstalk 控制台为网站配置文档根目录。

配置网站的文档根目录

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 对于 Document root（文档根目录），输入 **`/public`**。
6. 要保存更改，请选择页面底部的 Apply（应用）。
7. 更新完成后，单击 URL 以在浏览器中重新打开站点。

到目前为止，一切正常。接下来，您将在环境中添加一个数据库，并将 Laravel 配置为连接到该数据库。

将数据库添加到环境

在 Elastic Beanstalk 环境中启动 RDS 数据库实例。您可在 Elastic Beanstalk 上通过 Laravel 使用 MySQL、SQLServer 或 PostgreSQL 数据库。在本示例中，我们将使用 MySQL。

将 RDS 数据库实例添加到 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 对于引擎，选择 mysql。
6. 键入主 username（用户名）和 password（密码）。Elastic Beanstalk 将使用环境属性为应用程序提供这些值。
7. 要保存更改，请选择页面底部的 Apply（应用）。

创建数据库实例需要大约 10 分钟。有关数据库与 Elastic Beanstalk 环境耦合的更多信息，请参阅将数据库添加到 Elastic Beanstalk 环境。

同时，您可更新源代码，以便从环境中读取连接信息。Elastic Beanstalk 使用 `RDS_HOSTNAME` 等环境变量提供连接详细信息，以便您可以从应用程序中访问。

Laravel 的数据库配置存储在项目代码中 `database.php` 文件夹下名为 `config` 的文件内。查找 `mysql` 条目并修改 `host`、`database`、`username` 和 `and password` 变量，以便从 Elastic Beanstalk 中读取对应的值：

Example ~/Eb-laravel/config/database.php

```
...
    'connections' => [
```

```
        'sqlite' => [
            'driver' => 'sqlite',
            'database' => env('DB_DATABASE', database_path('database.sqlite')),
            'prefix' => '',
        ],

        'mysql' => [
            'driver' => 'mysql',
            'host' => env('RDS_HOSTNAME', '127.0.0.1'),
            'port' => env('RDS_PORT', '3306'),
            'database' => env('RDS_DB_NAME', 'forge'),
            'username' => env('RDS_USERNAME', 'forge'),
            'password' => env('RDS_PASSWORD', ''),
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'strict' => true,
            'engine' => null,
        ],
...
```

要验证数据库连接是否已正确配置，请向 index.php 添加代码以连接到数据库，并向默认响应添加一些代码：

Example ~/Eb-laravel/public/index.php

```
...
if(DB::connection()->getDatabaseName())
{
   echo "Connected to database ".DB::connection()->getDatabaseName();
}
$response->send();
...
```

在数据库实例完成启动后，将已更新的应用程序打包并将其部署到环境。

更新 Elastic Beanstalk 环境

1. 创建源包：

```
~/eb-laravel$ zip ../laravel-v2-rds.zip -r * .[^.]* -x "vendor/*"
```

2. 打开 Elastic Beanstalk 控制台，然后在“区域”列表中，选择您的。 AWS 区域
3. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

4. 选择上传和部署。
5. 选择 Browse（浏览），然后上传 `laravel-v2-rds.zip`。
6. 选择部署。

部署应用程序的新版本花费不到 1 分钟的时间。完成部署后，再次刷新网页，以验证数据库连接成功：

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在“区域”列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。

4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

此外，您还可以终止在您的 Elastic Beanstalk 环境外部创建的数据库资源。当您终止 Amazon RDS 数据库实例时，可以拍摄快照并在以后将数据恢复到其他实例。

终止您的 RDS 数据库实例

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例。
4. 选择操作，然后选择删除。
5. 选择是否创建快照，然后选择删除。

后续步骤

有关 Laravel 的更多信息，请访问 Laravel 官方网站 laravel.com。

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

在本教程中，您使用了 Elastic Beanstalk 控制台来配置 Composer 选项。要使此配置成为应用程序源的一部分，您可以使用类似于下面的配置文件。

Example .ebextensions/composer.config

```
option_settings:
  aws:elasticbeanstalk:container:php:phpini:
    document_root: /public
```

有关更多信息，请参阅 使用配置文件 (.ebextensions) 进行高级环境自定义。

在 Elastic Beanstalk 环境中运行 Amazon RDS 数据库实例不仅适合开发和测试，还将数据库的生命周期与环境相关联。请参阅向 PHP 应用程序环境中添加 Amazon RDS 数据库实例，浏览有关连接到在环境外运行的数据库的说明。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

将 CakePHP 应用程序部署到 Elastic Beanstalk

CakePHP 是用于 PHP 的开放源 MVC 框架。本教程将指导您完成生成 CakePHP 项目、将该项目部署到 Elastic Beanstalk 环境并配置该项目以连接到 Amazon RDS 数据库实例的过程。

Sections

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

CakePHP 4 需要 PHP 7.2 或更高版本。它还需要 CakePHP 官方安装文档中列出的 PHP 扩展。按照设置 PHP 开发环境主题中的说明安装 PHP 和 Composer。

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 PHP 平台并接受默认设置和示例代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &Environmentype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到 公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

安装 CakePHP 并生成网站

Composer 可使用一个命令安装 CakePHP 并创建工作项目：

```
~$ composer create-project --prefer-dist cakephp/app eb-cake
```

Composer 安装 CakePHP 及其约 20 个依赖项，并生成默认项目。

如果您在安装 CakePHP 时遇到问题，请访问官方文档中的安装主题：http://book.cakephp.org/4.0/en/installation.html

部署您的应用程序

创建包含由 Composer 创建的文件的源包。以下命令将创建名为 `cake-default.zip` 的源包。它将排除 `vendor` 文件夹中的文件，因为这些文件会占用大量空间并且对于将您的应用程序部署到 Elastic Beanstalk 不是必需的。

```
eb-cake zip ../cake-default.zip -r * .[^.]* -x "vendor/*"
```

将源包上传到 Elastic Beanstalk 以将 CakePHP 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

> **Note**
>
> 为了进一步优化源包，请初始化一个 Git 存储库并使用 `git archive` 命令创建源包。默认 Symfony 项目包含一个 `.gitignore` 文件，该文件指示 Git 排除 `vendor` 文件夹以及部署时不需要的其他文件。

该过程完成时，单击 URL，在浏览器中打开 CakePHP 应用程序。

到目前为止，一切正常。接下来，您将在环境中添加一个数据库，并将 CakePHP 配置为连接到该数据库。

将数据库添加到环境

在 Elastic Beanstalk 环境中启动 Amazon RDS 数据库实例。您可在 Elastic Beanstalk 上通过 CakePHP 使用 MySQL、SQLServer 或 PostgreSQL 数据库。在本示例中，我们将使用 PostgreSQL。

将 Amazon RDS 数据库实例添加到 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Database（数据库）下，选择 Edit（编辑）。
5. 对于数据库引擎，请选择 postgres。
6. 键入主 username（用户名）和 password（密码）。Elastic Beanstalk 将使用环境属性为应用程序提供这些值。
7. 要保存更改，请选择页面底部的 Apply（应用）。

创建数据库实例需要大约 10 分钟。同时，您可更新源代码，以便从环境中读取连接信息。Elastic Beanstalk 使用 RDS_HOSTNAME 等环境变量提供连接详细信息，以便您可以从应用程序中访问。

CakePHP 的数据库配置位于项目代码中 app.php 文件夹下名为 config 的文件内。打开此文件，添加从 $_SERVER 读取环境变量的代码，并将其分配到本地变量。在第一行 (<?php) 后插入以下示例中突出显示的行：

Example ~/Eb-cake/config/app.php

```
<?php
if (!defined('RDS_HOSTNAME')) {
  define('RDS_HOSTNAME', $_SERVER['RDS_HOSTNAME']);
  define('RDS_USERNAME', $_SERVER['RDS_USERNAME']);
  define('RDS_PASSWORD', $_SERVER['RDS_PASSWORD']);
  define('RDS_DB_NAME', $_SERVER['RDS_DB_NAME']);
}
return [
...
```

数据库连接可在 app.php 中进行配置。使用与数据库引擎（Mysql、Sqlserver 或 Postgres）相匹配的驱动程序名称，查找以下部分并修改默认数据源配置，并设置 host、username、password 和 database 变量，以便从 Elastic Beanstalk 读取相应的值：

Example ~/Eb-cake/config/app.php

```
...
```

```
     /**
     * Connection information used by the ORM to connect
     * to your application's datastores.
     * Drivers include Mysql Postgres Sqlite Sqlserver
     * See vendor\cakephp\cakephp\src\Database\Driver for complete list
     */
    'Datasources' => [
        'default' => [
            'className' => 'Cake\Database\Connection',
            'driver' => 'Cake\Database\Driver\Postgres',
            'persistent' => false,
            'host' => RDS_HOSTNAME,
            /*
             * CakePHP will use the default DB port based on the driver selected
             * MySQL on MAMP uses port 8889, MAMP users will want to uncomment
             * the following line and set the port accordingly
             */
            //'port' => 'non_standard_port_number',
            'username' => RDS_USERNAME,
            'password' => RDS_PASSWORD,
            'database' => RDS_DB_NAME,
            /*
             * You do not need to set this flag to use full utf-8 encoding (internal
 default since CakePHP 3.6).
             */
            //'encoding' => 'utf8mb4',
            'timezone' => 'UTC',
            'flags' => [],
            'cacheMetadata' => true,
            'log' => false,
...
```

在数据库实例完成启动时，请绑定已更新的应用程序并将其部署于环境：

更新 Elastic Beanstalk 环境

1. 创建源包：

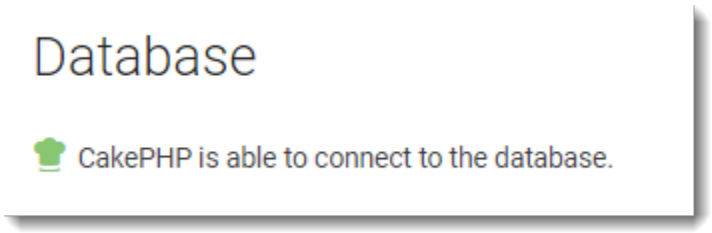
   ```
   ~/eb-cake$ zip ../cake-v2-rds.zip -r * .[^.]* -x "vendor/*"
   ```

2. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
3. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

4. 选择上传和部署。
5. 选择 Browse（浏览），然后上传 `cake-v2-rds.zip`。
6. 选择部署。

部署应用程序的新版本花费不到 1 分钟的时间。完成部署后，再次刷新网页，以验证数据库连接成功：



清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

此外，您还可以终止在您的 Elastic Beanstalk 环境外部创建的数据库资源。当您终止 Amazon RDS 数据库实例时，可以拍摄快照并在以后将数据恢复到其他实例。

终止您的 RDS 数据库实例

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例。
4. 选择操作，然后选择删除。
5. 选择是否创建快照，然后选择删除。

后续步骤

如需了解有关 CakePHP 的更多信息，请访问 book.cakephp.org，阅读书籍。

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

在 Elastic Beanstalk 环境中运行 Amazon RDS 数据库实例不仅适合开发和测试，还将数据库的生命周期与环境相关联。请参阅向 PHP 应用程序环境中添加 Amazon RDS 数据库实例，浏览有关连接到在环境外运行的数据库的说明。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

将 Symfony 应用程序部署到 Elastic Beanstalk

Symfony 是一个开源框架，用于开发动态 PHP Web 应用程序。本教程将引导您完成生成 Symfony 应用程序并将其部署到环境的 AWS Elastic Beanstalk 过程。

Sections

- 后续步骤

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

Symfony 4.4.9 需要 PHP 7.1.3 或更高版本。它还需要 Symfony 官方安装文档中的技术要求主题中列出的 PHP 扩展。在本教程中，我们使用 PHP 7.2 和相应的 Elastic Beanstalk 平台版本。按照设置 PHP 开发环境主题中的说明安装 PHP 和 Composer。

如需 Symfony 支持和维护信息，请参阅 Symfony 网站上的 Symfony 发行主题。有关 Symfony 4.4.9 的 PHP 版本支持相关更新的更多信息，请参阅 Symfony 网站上的 Symfony 4.4.9 发行说明主题。

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 PHP 平台并接受默认设置和示例代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

安装 Symfony 并生成网站

Composer 可使用一个命令安装 Symfony 并创建工作项目：

```
~$ composer create-project symfony/website-skeleton eb-symfony
```

Composer 将安装 Symfony 及其依赖项，并生成默认项目。

如果在安装 Symfony 时遇到任何问题，请参阅 Symfony 官方文档中的安装主题。

部署您的应用程序

转到项目目录。

```
~$ cd eb-symfony
```

创建包含由 Composer 创建的文件的源包。以下命令将创建名为 `symfony-default.zip` 的源包。它将排除 `vendor` 文件夹中的文件，因为这些文件会占用大量空间并且对于将您的应用程序部署到 Elastic Beanstalk 不是必需的。

```
eb-symfony$ zip ../symfony-default.zip -r * .[^.]* -x "vendor/*"
```

将源包上传到 Elastic Beanstalk 以将 Symfony 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

> **Note**
>
> 为了进一步优化源包，请初始化一个 Git 存储库并使用 `git archive` 命令创建源包。默认 Symfony 项目包含一个 `.gitignore` 文件，该文件指示 Git 排除 `vendor` 文件夹以及部署时不需要的其他文件。

配置 Composer 设置

部署完成后，单击 URL 以在浏览器中打开 Symfony 应用程序。

这是什么? 默认情况下，Elastic Beanstalk 提供项目在网站根路径下的根目录。但在这种情况下，默认页面 (`app.php`) 位于 `web` 文件夹的下一级。您可通过将 `/public` 添加至 URL 加以验证。例如，`http://symfony.us-east-2.elasticbeanstalk.com/public`。

要在根路径中提供 Symfony 应用程序，请使用 Elastic Beanstalk 控制台为网站配置文档根目录。

配置网站的文档根目录

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 对于 Document root（文档根目录），输入 **`/public`**。
6. 要保存更改，请选择页面底部的 Apply（应用）。
7. 更新完成后，单击 URL 以在浏览器中重新打开站点。

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

后续步骤

有关 Symfony 的更多信息，请参阅 symfony.com 上的什么是 Symfony？。

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

在本教程中，您使用了 Elastic Beanstalk 控制台来配置 Composer 选项。要使此配置成为应用程序源的一部分，您可以使用类似于下面的配置文件。

Example .ebextensions/composer.config

```
option_settings:
  aws:elasticbeanstalk:container:php:phpini:
    document_root: /public
```

有关更多信息，请参阅 使用配置文件 (.ebextensions) 进行高级环境自定义。

Symfony 使用自己的配置文件来配置数据库连接。有关使用 Symfony 连接到数据库的说明，请参阅使用 Symfony 连接到数据库。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

将带有外部 Amazon RDS 数据库的高可用性 PHP 应用程序部署到 Elastic Beanstalk

本教程将引导您完成在外部启动 RDS 数据库实例 AWS Elastic Beanstalk，以及配置运行 PHP 应用程序的高可用性环境以连接到该实例的过程。运行 Elastic Beanstalk 外部的数据库实例会将数据库与环境的生命周期分离。这让您可以从多个环境连接到同一个数据库，将一个数据库交换为另一个数据库，或执行蓝/绿部署，而不影响您的数据库。

本教程使用了一个示例 PHP 应用程序，该应用程序利用 MySQL 数据库来存储用户提供的文本数据。示例应用程序使用配置文件配置 PHP 设置并为要使用的应用程序在数据库中创建表。本教程还展示了如何使用 Composer 文件在部署期间安装程序包。

Sections

先决条件

在开始之前，请从以下地址下载示例应用程序源包 GitHub：eb-demo-php-simple-app- 1.3.zip

本教程中针对 Amazon Relational Database Service (Amazon RDS) 任务的过程假定您正在默认的 Amazon Virtual Private Cloud (Amazon VPC) 中启动资源。所有新账户在每个区域中均包含一个默认 VPC。如果您没有默认 VPC，则这些过程将会发生变化。有关 EC2-Classic 和自定义 VPC 平台的说明，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

在 Amazon RDS 中启动数据库实例

要将外部数据库与在 Elastic Beanstalk 中运行的应用程序结合使用，请首先使用 Amazon RDS 启动数据库实例。当您使用 Amazon RDS 启动实例时，它完全独立于 Elastic Beanstalk 和您的 Elastic Beanstalk 环境，并且不会由 Elastic Beanstalk 终止或监控。

使用 Amazon RDS 控制台可启动多可用区 MySQL 数据库实例。选择多可用区部署可确保您的数据库将进行故障转移并在源数据库实例中断服务时继续可用。

在默认 VPC 中启动 RDS 数据库实例

1. 打开 RDS 控制台。
2. 在导航窗格中，选择 Databases（数据库）。
3. 选择创建数据库。
4. 选择 Standard Create（标准创建）。

> **Important**
>
> 请勿选择 Easy Create（轻松创建）。如果您选择它，您将无法配置启动此 RDS 数据库所需的设置。

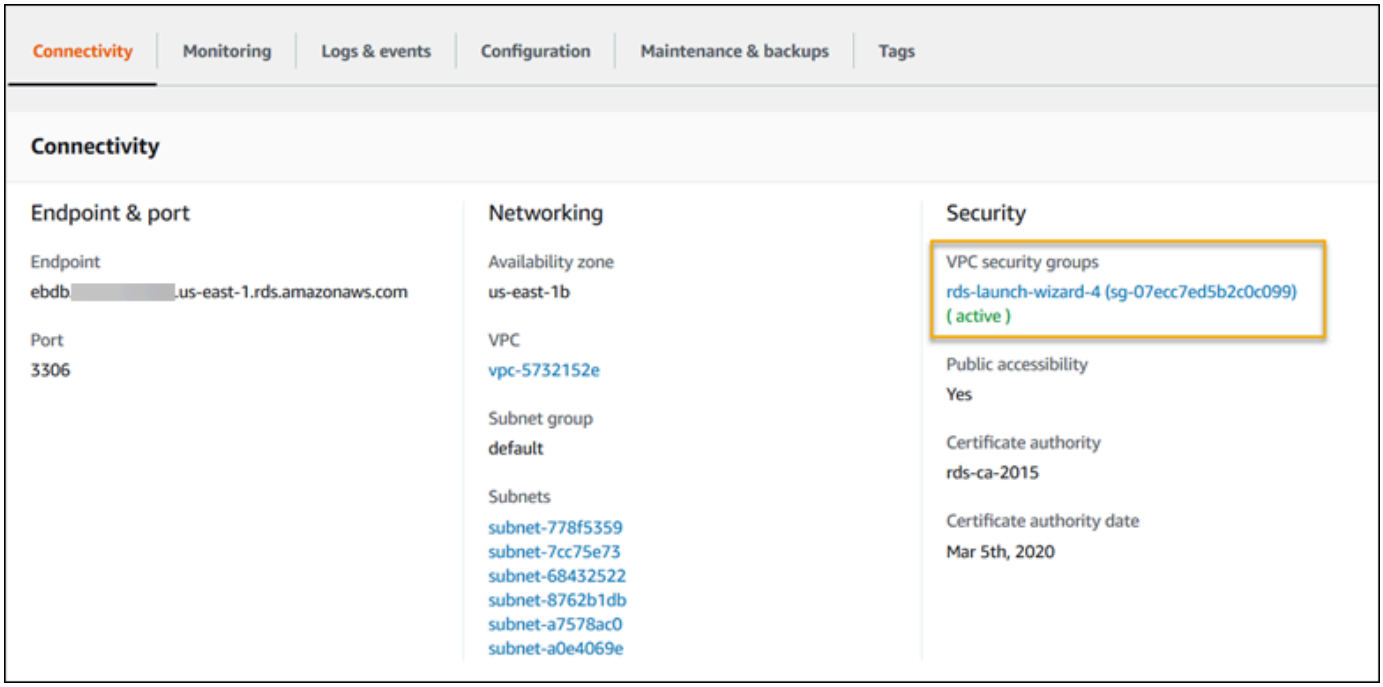
5. 在 Additional configuration（附加配置）下，对于 Initial database name（初始数据库名称），键入 **ebdb**。
6. 请查看默认设置并根据您的具体要求调整这些设置。请注意以下选项：
   - DB instance class（数据库实例类）– 选择对于您的工作负载具有适当的内存量和 CPU 能力的实例大小。
   - Multi-AZ deployment（多可用区部署）– 为了实现高可用性，请将其设置为 Create an Aurora Replica/Reader node in a different AZ（在不同的可用区中创建 Aurora 副本/读取器节点）。
   - Master username（主用户名）和 Master password（主密码）– 数据库用户名和密码。请记下这些设置，因为您以后将使用这些值。
7. 验证其余选项的默认设置，然后选择 Create database（创建数据库）。

接下来，修改附加到数据库实例的安全组以允许相应的端口上的入站流量。这与您稍后要附加到 Elastic Beanstalk 环境的安全组相同，因此您添加的规则将向同一安全组中的其他资源授予入口流量权限。

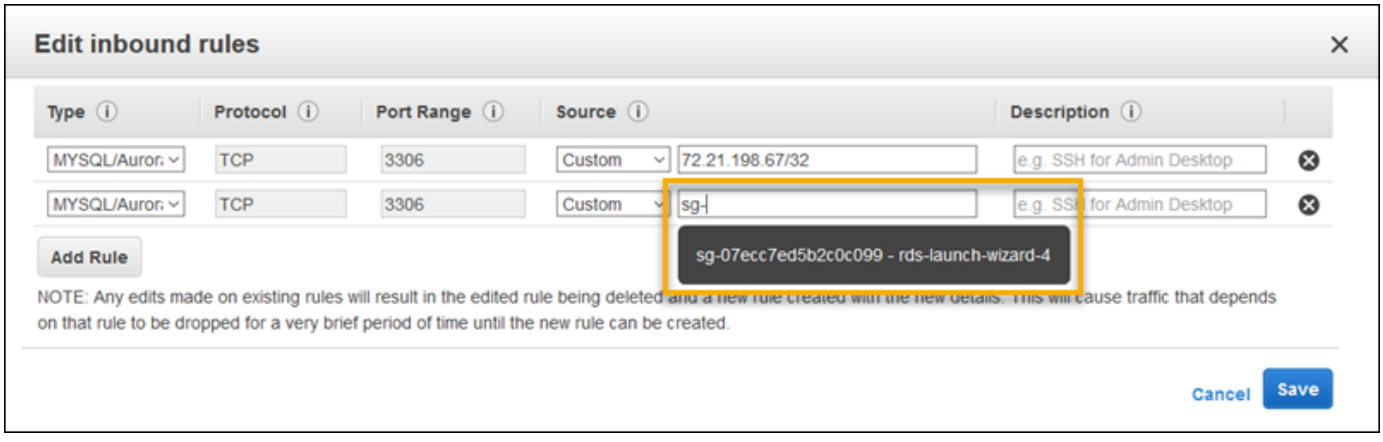
修改附加到 RDS 实例的安全组上的入站规则

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例的名称以查看其详细信息。

4. 在 Connectivity（连接）部分中，记下在该页上显示的 Subnets（子网）、Security groups（安全组）和 Endpoint（端点）。这样您稍后可以使用这些信息。
5. 在 Security（安全性）下面，您可以查看与数据库实例关联的安全组。打开链接以在 Amazon EC2 控制台中查看安全组。



6. 在安全组详细信息中，选择 Inbound（入站）选项卡。
7. 选择 Edit（编辑）。
8. 选择 Add Rule（添加规则）。
9. 对于 Type（类型），请选择应用程序使用的数据库引擎。
10. 对于 Source（源），键入 **sg-** 以查看可用安全组的列表。请选择 Elastic Beanstalk 环境中使用的与 Auto Scaling 组关联的安全组。这样，环境中的 Amazon EC2 实例就可以访问数据库。



11. 选择 Save（保存）。

创建一个数据库实例大约需要 10 分钟。与此同时，创建您的 Elastic Beanstalk 环境。

创建 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 PHP 平台并接受默认设置和示例代码。启动环境后，您可以将环境配置为连接到数据库，然后部署从中下载的示例应用程序 GitHub。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。

- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。您启动的 RDS 数据库实例位于您的环境之外，因此您需负责管理其生命周期。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

配置安全组、环境属性和扩展

将数据库实例的安全组添加到正在运行的环境。此过程将使 Elastic Beanstalk 使用附加的其他安全组重新配置环境中的所有实例。

向环境添加安全组

- 请执行以下操作之一：
  - 使用 Elastic Beanstalk 控制台添加安全组

    a. 打开 Elastic Beanstalk 控制台，然后在“区域”列表中，选择您的。 AWS 区域

    b. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

c. 在导航窗格中，选择 Configuration（配置）。

d. 在 Instances（实例）配置类别中，选择 Edit（编辑）。

e. 在 EC2 security groups（EC2 安全组）下面，选择要附加到实例的安全组以及 Elastic Beanstalk 创建的实例安全组。

f. 要保存更改，请选择页面底部的 Apply（应用）。

g. 阅读警告，然后选择 Confirm（确认）。

- 要使用配置文件添加安全组，请使用 `securitygroup-addexisting.config` 示例文件。

接下来，使用环境属性将连接信息传递给环境。示例应用程序使用了默认的一组属性，这些属性与您在环境中预置数据库时 Elastic Beanstalk 配置的属性匹配。

为 Amazon RDS 数据库实例配置环境属性

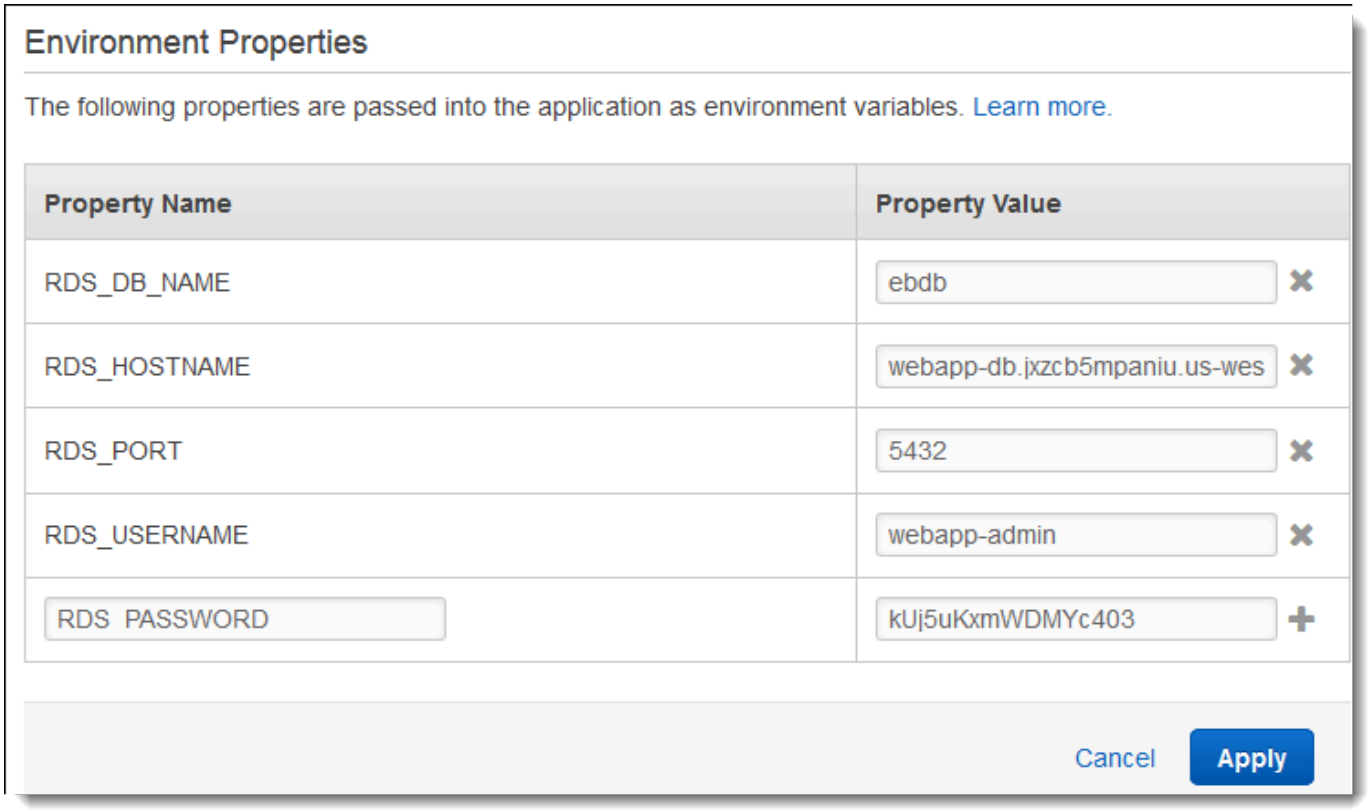
1. 打开 Elastic Beanstalk 控制台，然后在“区域”列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在环境属性部分中，定义应用程序读取的用于构建连接字符串的变量。为了实现与具有集成 RDS 数据库实例的环境的兼容，请使用以下名称和值。您可以在 RDS 控制台中找到除密码以外的所有值。

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity &

属性名称	描述	属性值
		security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。



6. 要保存更改，请选择页面底部的 Apply（应用）。

最后，使用较高的最低实例计数配置您环境的 Auto Scaling 组。请始终至少运行两个实例，以防止您环境中的 Web 服务器发生单点故障，并支持您在不中断站点服务的情况下部署更改。

配置您的环境的 Auto Scaling 组以获得高可用性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在 Auto Scaling group（Auto Scaling 组）部分中，将 Min instances（最小实例数）设置为 **2**。

6. 要保存更改，请选择页面底部的 Apply（应用）。

部署示例应用程序。

现在，您的环境已准备好运行示例应用程序并连接到 Amazon RDS。将示例应用程序部署到您的环境。

> **Note**
>
> 如果你还没有 GitHub，可以从以下网址下载源包：eb-demo-php-simple-app- 1.3.zip

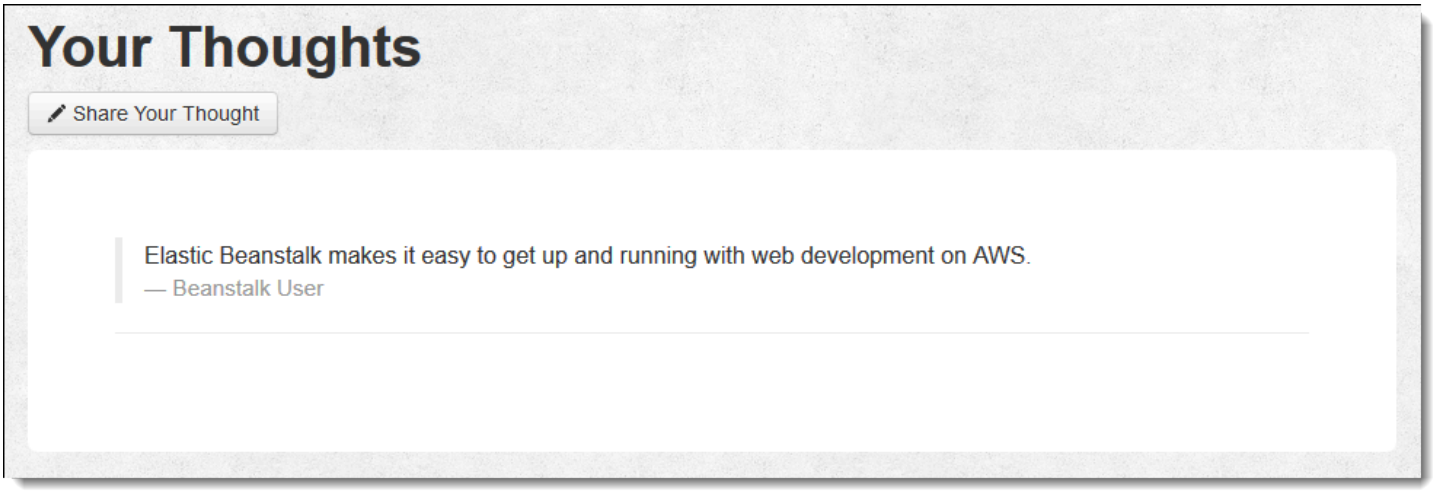
部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

此站点将收集用户评论并使用 MySQL 数据库来存储数据。要添加评论，请选择 Share Your Thought（分享您的想法），输入评论，然后选择 Submit Your Thought（提交您的想法）。Web 应用程序会将评论写入数据库，以便环境中的任何实例均可读取它，并且在实例中断服务时不会丢失。



清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

此外，您还可以终止在您的 Elastic Beanstalk 环境外部创建的数据库资源。当您终止 Amazon RDS 数据库实例时，可以拍摄快照并在以后将数据恢复到其他实例。

终止您的 RDS 数据库实例

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例。

4. 选择操作，然后选择删除。
5. 选择是否创建快照，然后选择删除。

后续步骤

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

示例应用程序使用配置文件配置 PHP 设置并在数据库中创建表 (如果表尚未存在)。您还可以使用配置文件在环境创建期间配置实例的安全组设置，以避免耗时的配置更新。请参阅使用配置文件 (.ebextensions) 进行高级环境自定义了解更多信息。

对于开发和测试，您可能希望使用 Elastic Beanstalk 的用于将托管数据库实例直接添加到环境的功能。有关在环境内设置数据库的说明，请参阅将数据库添加到 Elastic Beanstalk 环境。

如果您需要高性能数据库，请考虑使用 Amazon Aurora。Amazon Aurora 是与 MySQL 兼容的数据库引擎，能够以低成本提供商用数据库功能。要将您的应用程序连接到另一个数据库，请重复执行安全组配置步骤并更新与 RDS 相关的环境属性。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

将带有外部 Amazon RDS 数据库的高可用性 WordPress 网站部署到 Elastic Beanstalk

本教程介绍如何启动外部的 Amazon RDS 数据库实例 AWS Elastic Beanstalk，以及如何配置运行 WordPress 网站的高可用性环境以连接到该实例。该网站使用 Amazon Elastic File System (Amazon EFS) 作为已上传文件的共享存储空间。

运行 Elastic Beanstalk 外部的数据库实例会将数据库与环境的生命周期分离。这让您可以从多个环境连接到同一个数据库，将一个数据库交换为另一个数据库，或执行蓝/绿部署，而不影响您的数据库。

> **Note**
>
> 有关 PHP WordPress 版本与版本兼容性的最新信息，请参阅 WordPress 网站上的 PHP 兼容性和 WordPress 版本。在为 WordPress 实现升级到新版本的 PHP 之前，您应该参考这些信息。

主题

- 先决条件
- 在 Amazon RDS 中启动数据库实例
- 下载 WordPress
- 启动 Elastic Beanstalk 环境
- 配置安全组和环境属性
- 配置并部署您的应用程序
- 安装 WordPress
- 更新密钥和加密盐
- 删除访问限制
- 配置 Auto Scaling 组
- 升级 WordPress
- 清理
- 后续步骤

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

默认 VPC

本教程中的 Amazon Relational Database Service (Amazon RDS) 过程假定您在默认 Amazon Virtual Private Cloud (Amazon VPC) 中启动资源。所有新账户在每个 AWS 区域都包含一个默认 VPC。如果您没有默认 VPC，则这些过程将会发生变化。有关 EC2-Classic 和自定义 VPC 平台的说明，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

AWS 区域

该示例应用程序使用 Amazon EFS，它仅适用于支持 Amazon EFS 的 AWS 区域。要了解支持的 AWS 区域，请参阅中的 Amazon Elastic File System 终端节点和配额AWS 一般参考。

在 Amazon RDS 中启动数据库实例

当您使用 Amazon RDS 启动实例时，它完全独立于 Elastic Beanstalk 和您的 Elastic Beanstalk 环境，并且不会由 Elastic Beanstalk 终止或监控。

在以下步骤中，您将使用 Amazon RDS 控制台：

- 使用 MySQL 引擎启动数据库。
- 启用多可用区部署。这会在其他可用区 (AZ) 中创建备用数据库，以提供数据冗余、消除 I/O 冻结并最大限度地减少系统备份期间的延迟峰值。

在默认 VPC 中启动 RDS 数据库实例

1. 打开 RDS 控制台。
2. 在导航窗格中，选择 Databases（数据库）。
3. 选择创建数据库。
4. 选择 Standard Create（标准创建）。

> **Important**
>
> 请勿选择 Easy Create（轻松创建）。如果您选择它，您将无法配置启动此 RDS 数据库所需的设置。

5. 在 Additional configuration（附加配置）下，对于 Initial database name（初始数据库名称），键入 **ebdb**。
6. 请查看默认设置并根据您的具体要求调整这些设置。请注意以下选项：

   - DB instance class（数据库实例类）– 选择对于您的工作负载具有适当的内存量和 CPU 能力的实例大小。
   - Multi-AZ deployment（多可用区部署）– 为了实现高可用性，请将其设置为 Create an Aurora Replica/Reader node in a different AZ（在不同的可用区中创建 Aurora 副本/读取器节点）。
   - Master username（主用户名）和 Master password（主密码）– 数据库用户名和密码。请记下这些设置，因为您以后将使用这些值。

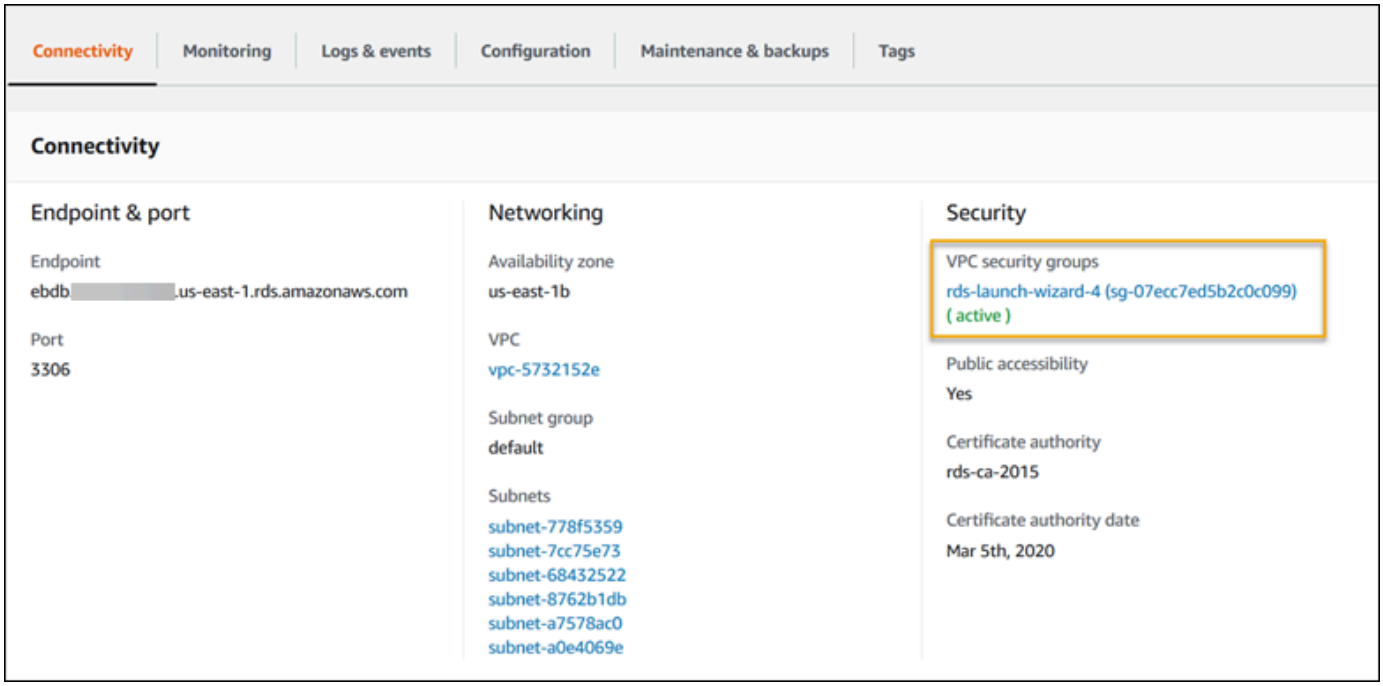
7. 验证其余选项的默认设置，然后选择 Create database（创建数据库）。

创建数据库实例后，修改附加到该实例的安全组，以便允许相应端口上的入站流量。

> **Note**
>
> 这与您稍后要附加到 Elastic Beanstalk 环境的安全组相同，因此您添加的规则现在将向同一安全组中的其他资源授予入口流量权限。

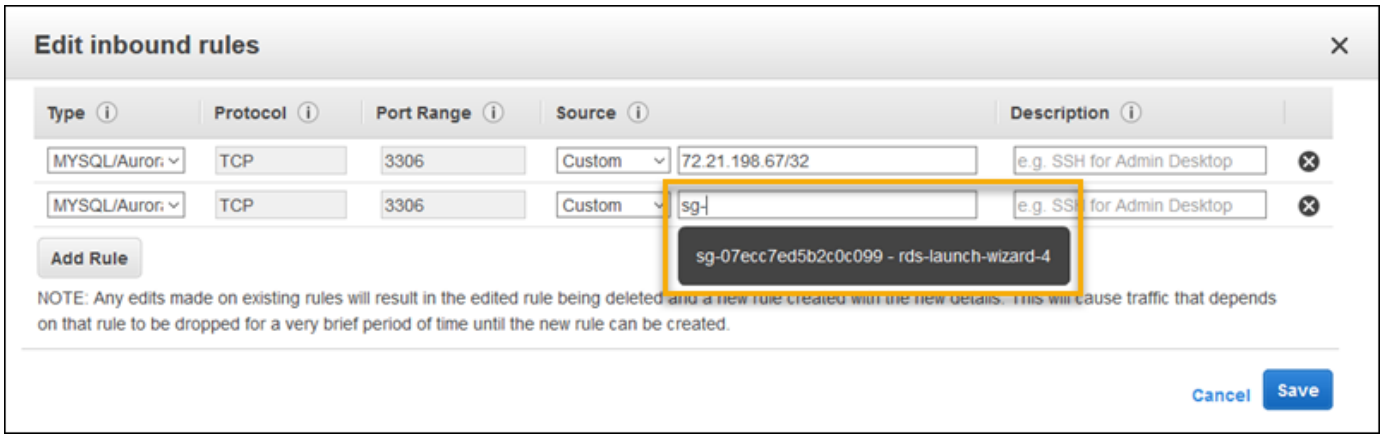
修改附加到 RDS 实例的安全组上的入站规则

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例的名称以查看其详细信息。
4. 在 Connectivity（连接）部分中，记下在该页上显示的 Subnets（子网）、Security groups（安全组）和 Endpoint（端点）。这样您稍后可以使用这些信息。
5. 在 Security（安全性）下面，您可以查看与数据库实例关联的安全组。打开链接以在 Amazon EC2 控制台中查看安全组。



6. 在安全组详细信息中，选择 Inbound（入站）选项卡。
7. 选择 Edit（编辑）。

8. 选择 Add Rule（添加规则）。
9. 对于 Type（类型），请选择应用程序使用的数据库引擎。
10. 对于 Source（源），键入 **sg-** 以查看可用安全组的列表。请选择 Elastic Beanstalk 环境中使用的与 Auto Scaling 组关联的安全组。这样，环境中的 Amazon EC2 实例就可以访问数据库。



11. 选择 Save（保存）。

创建一个数据库实例大约需要 10 分钟。同时，下载 WordPress 并创建你的 Elastic Beanstalk 环境。

下载 WordPress

要准备 WordPress 使用进行部署 AWS Elastic Beanstalk，必须将 WordPress 文件复制到您的计算机并提供正确的配置信息。

创建 WordPress 项目

1. WordPress 从 wordpress.org 下载。

```
~$curl https://wordpress.org/wordpress-6.2.tar.gz -o wordpress.tar.gz
```

2. 从示例存储库中下载配置文件。

```
~$ wget https://github.com/aws-samples/eb-php-wordpress/releases/download/v1.1/eb-
php-wordpress-v1.zip
```

3. 提取 WordPress 并更改文件夹的名称。

```
~$ tar -xvf wordpress.tar.gz
~$ mv wordpress wordpress-beanstalk
~$ cd wordpress-beanstalk
```

4. 在 WordPress 安装过程中解压缩配置文件。

```
~/wordpress-beanstalk$ unzip ../eb-php-wordpress-v1.zip
 creating: .ebextensions/
inflating: .ebextensions/dev.config
inflating: .ebextensions/efs-create.config
inflating: .ebextensions/efs-mount.config
inflating: .ebextensions/loadbalancer-sg.config
inflating: .ebextensions/wordpress.config
inflating: LICENSE
inflating: README.md
inflating: wp-config.php
```

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。启动环境后，您可以将其配置为连接到数据库，然后将 WordPress 代码部署到环境中。

在以下步骤中，您将使用 Elastic Beanstalk 控制台：

- 使用托管的 PHP 平台创建 Elastic Beanstalk 应用程序。
- 接受默认设置和示例代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

创建环境大约需要 5 分钟，将创建以下资源。

Elastic Beanstalk 创建的资源

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。

- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

由于您启动的 Amazon RDS 实例位于您的环境之外，因此您需负责管理其生命周期。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

配置安全组和环境属性

将数据库实例的安全组添加到正在运行的环境。此过程将使 Elastic Beanstalk 使用附加的其他安全组重新配置环境中的所有实例。

向环境添加安全组

- 请执行以下操作之一：
  - 使用 Elastic Beanstalk 控制台添加安全组
    a. 打开 Elastic Beanstalk 控制台，然后在“区域”列表中，选择您的。 AWS 区域
    b. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

    > **Note**
    >
    > 如果您有多个环境，请使用搜索栏筛选环境列表。

    c. 在导航窗格中，选择 Configuration（配置）。
    d. 在 Instances（实例）配置类别中，选择 Edit（编辑）。
    e. 在 EC2 security groups（EC2 安全组）下面，选择要附加到实例的安全组以及 Elastic Beanstalk 创建的实例安全组。
    f. 要保存更改，请选择页面底部的 Apply（应用）。
    g. 阅读警告，然后选择 Confirm（确认）。
  - 要使用配置文件添加安全组，请使用 `securitygroup-addexisting.config` 示例文件。

接下来，使用环境属性将连接信息传递给环境。

该 WordPress 应用程序使用一组默认属性，这些属性与 Elastic Beanstalk 在环境中配置数据库时配置的属性相匹配。

为 Amazon RDS 数据库实例配置环境属性

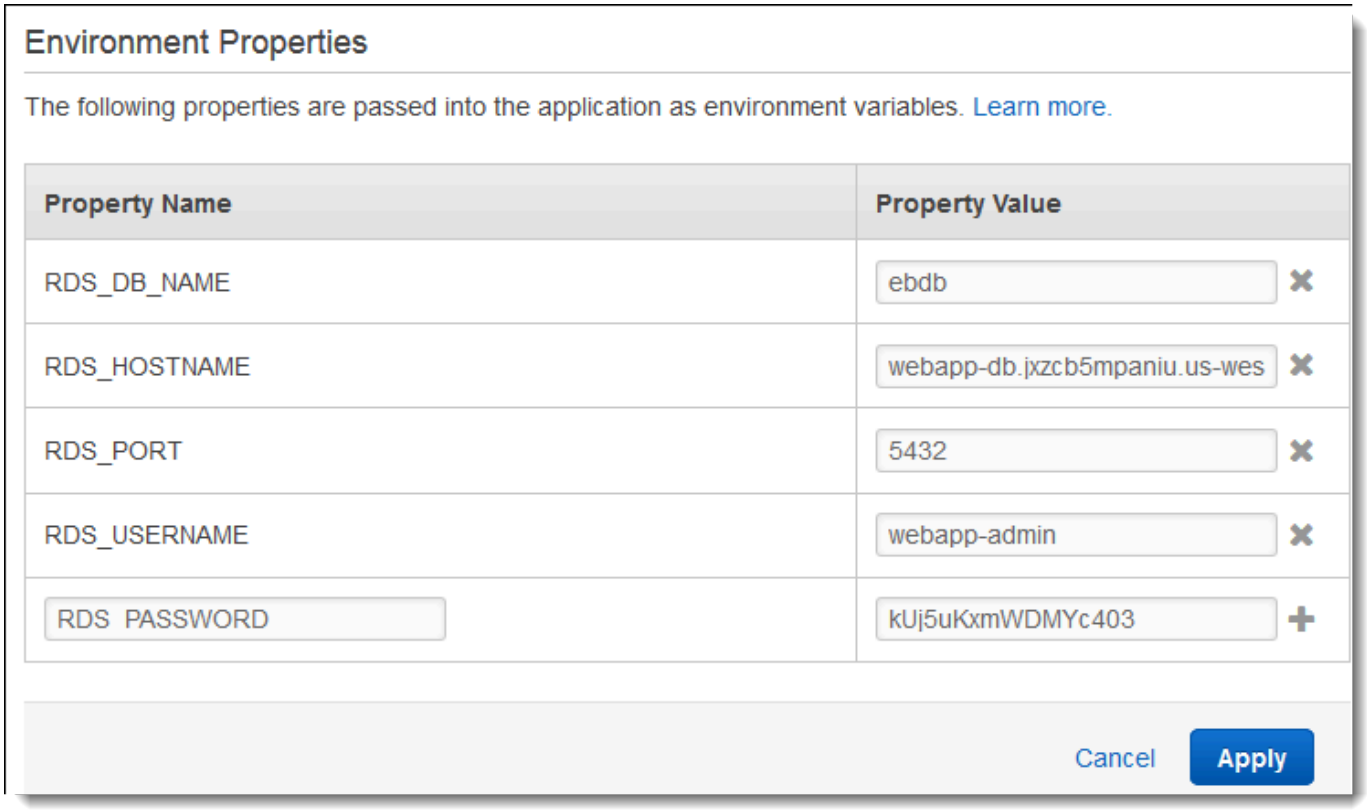
1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在环境属性部分中，定义应用程序读取的用于构建连接字符串的变量。为了实现与具有集成 RDS 数据库实例的环境的兼容，请使用以下名称和值。您可以在 RDS 控制台中找到除密码以外的所有值。

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。

属性名称	描述	属性值
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。



6. 要保存更改，请选择页面底部的 Apply（应用）。

配置并部署您的应用程序

验证您的 wordpress-beanstalk 文件夹结构是否正确，如下所示。

```
wordpress-beanstalk$ tree -aL 1
.
### .ebextensions
### index.php
### LICENSE
### license.txt
### readme.html
### README.md
### wp-activate.php
```

```
### wp-admin
### wp-blog-header.php
### wp-comments-post.php
### wp-config.php
### wp-config-sample.php
### wp-content
### wp-cron.php
### wp-includes
### wp-links-opml.php
### wp-load.php
### wp-login.php
### wp-mail.php
### wp-settings.php
### wp-signup.php
### wp-trackback.php
### xmlrpc.php
```

项目存储库中的自定义 `wp-config.php` 文件使用您在上一步中定义的环境变量来配置数据库连接。`.ebextensions` 文件夹包含用于在您的 Elastic Beanstalk 环境中创建其他资源的配置文件。

该配置文件要求您进行修改以使用账户。将文件中的占位符值替换为适当的 ID 并创建源代码包。

更新配置文件并创建源包

1. 按如下方式修改配置文件。

   - `.ebextensions/dev.config`— 限制对环境的访问以在 WordPress 安装过程中对其进行保护。将文件顶部附近的占位符 IP 地址替换为用于访问环境网站的计算机的公有 IP 地址，从而完成 WordPress 安装。

     > **Note**
     >
     > 根据您的网络，您可能需要使用 IP 地址块。

   - `.ebextensions/efs-create.config` - 在您的 VPC 中的每个可用区/子网中创建 EFS 文件系统和装载点。在 Amazon VPC 控制台中确定您的默认 VPC 和子网 ID。

2. 创建将文件包含在项目文件夹中的源代码包。以下命令将创建名为 `wordpress-beanstalk.zip` 的源包。

   ```
   ~/eb-wordpress$ zip ../wordpress-beanstalk.zip -r * .[^.]*
   ```

将源包上传到 Elastic Beanstalk WordPress 以部署到您的环境中。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

安装 WordPress

要完成 WordPress 安装

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 URL 以便在浏览器中打开您的站点。您将被重定向到 WordPress 安装向导，因为您尚未配置该站点。
4. 执行标准安装。`wp-config.php` 文件在源代码中已存在，并配置为从环境中读取数据库连接信息。系统不会提示您配置连接。

安装大约需要一分钟能完成。

更新密钥和加密盐

WordPress 配置文件`wp-config.php`还会从环境属性中读取键和盐的值。当前，这些属性全都被 `test` 文件夹中的 `wordpress.config` 文件设置为 `.ebextensions`。

哈希加密盐可以是符合环境属性要求的任意值，但不应将其存储在源代码控制中。使用 Elastic Beanstalk 控制台可直接针对环境设置这些属性。

更新环境属性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Software（软件）下，选择 Edit（编辑）。
5. 对于 `Environment properties`，修改以下属性：
   - `AUTH_KEY` - 为 `AUTH_KEY` 选择的值。
   - `SECURE_AUTH_KEY` - 为 `SECURE_AUTH_KEY` 选择的值。
   - `LOGGED_IN_KEY` - 为 `LOGGED_IN_KEY` 选择的值。
   - `NONCE_KEY` - 为 `NONCE_KEY` 选择的值。
   - `AUTH_SALT` - 为 `AUTH_SALT` 选择的值。
   - `SECURE_AUTH_SALT` - 为 `SECURE_AUTH_SALT` 选择的值。
   - `LOGGED_IN_SALT` - 为 `LOGGED_IN_SALT` 选择的值。
   - `NONCE_SALT` - 为 `NONCE_SALT` 选择的值。
6. 要保存更改，请选择页面底部的 Apply（应用）。

> **Note**
>
> 直接对环境设置属性会覆盖 `wordpress.config` 的值。

删除访问限制

示例项目包括配置文件 `loadbalancer-sg.config`。它会使用您在 `dev.config` 中配置的 IP 地址创建安全组，并将其分配给环境的负载均衡器。它将端口 80 上的 HTTP 访问限制为来自您的网络的连接。否则，在您安装 WordPress 和配置管理员帐户之前，外部人员可能会连接到您的站点。

现在您已经安装完毕 WordPress，请删除配置文件以向全世界开放该站点。

删除限制并更新环境

1. 从项目目录中删除 `.ebextensions/loadbalancer-sg.config` 文件。

```
~/wordpress-beanstalk$ rm .ebextensions/loadbalancer-sg.config
```

2. 创建源包。

```
~/eb-wordpress$ zip ../wordpress-beanstalk-v2.zip -r * .[^.]*
```

将源包上传到 Elastic Beanstalk WordPress 以部署到您的环境中。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

配置 Auto Scaling 组

最后，使用较高的最低实例计数配置您环境的 Auto Scaling 组。请始终运行至少两个实例，以防止您的环境中的 Web 服务器发生单点故障。这样，您还可以在不中断站点服务的情况下部署更改。

配置您的环境的 Auto Scaling 组以获得高可用性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在 Auto Scaling group（Auto Scaling 组）部分中，将 Min instances（最小实例数）设置为 **2**。
6. 要保存更改，请选择页面底部的 Apply（应用）。

为了支持多个实例中的内容上传，示例项目使用 Amazon EFS 来创建共享文件系统。在站点上创建一个帖子，然后上传内容以将其存储在共享文件系统上。查看帖子并多次刷新页面，以命中两个实例并验证共享文件系统是否正常工作。

升级 WordPress

要升级到的新版本 WordPress，请备份您的站点并将其部署到新环境。

> **Important**
>
> 请勿使用其中的更新功能， WordPress 也不要更新源文件以使用新版本。这两个操作都可能导致您的文章 URL 返回 404 错误，即使这些 URL 仍然位于数据库和文件系统中。

要升级 WordPress

1. 在 WordPress 管理员控制台中，使用导出工具将您的帖子导出到 XML 文件。
2. 使用与安装先前版本相同的步骤在 Elastic Beanstalk 上部署和安装新版本。 WordPress 要避免停机，您可以使用新版本创建环境。
3. 在新版本中，在管理员控制台中安装导入 WordPress 器工具，然后使用它来导入包含您的帖子的 XML 文件。如果文章由使用旧版本的管理员用户创建，请将文章分配到新站点上的管理员用户而不是尝试导入管理员用户。
4. 如果您将新版本部署到单独的环境，请执行 CNAME 交换 以将用户从旧站点重定向到新站点。

清理

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

此外，您还可以终止在您的 Elastic Beanstalk 环境外部创建的数据库资源。当您终止 Amazon RDS 数据库实例时，可以拍摄快照并在以后将数据恢复到其他实例。

终止您的 RDS 数据库实例

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例。
4. 选择操作，然后选择删除。
5. 选择是否创建快照，然后选择删除。

后续步骤

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

示例应用程序使用配置文件配置 PHP 设置，并在数据库中创建一个表 (如果尚未存在)。您还可以使用配置文件在环境创建期间配置实例的安全组设置，以避免耗时的配置更新。请参阅使用配置文件 (.ebextensions) 进行高级环境自定义了解更多信息。

对于开发和测试，您可能希望使用 Elastic Beanstalk 的用于将托管数据库实例直接添加到环境的功能。有关在环境内设置数据库的说明，请参阅将数据库添加到 Elastic Beanstalk 环境。

如果您需要高性能数据库，请考虑使用 Amazon Aurora。Amazon Aurora 是与 MySQL 兼容的数据库引擎，能够以低成本提供商用数据库功能。要将您的应用程序连接到另一个数据库，请重复执行安全组配置步骤并更新与 RDS 相关的环境属性。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

将带有外部 Amazon RDS 数据库的高可用性 Drupal 网站部署到 Elastic Beanstalk

本教程将引导您完成在外部启动 RDS 数据库实例的过程 AWS Elastic Beanstalk。然后，它介绍了配置运行 Drupal 网站的高可用性环境以连接到该数据库实例。该网站使用 Amazon Elastic File System (Amazon EFS) 作为已上传文件的共享存储空间。在 Elastic Beanstalk 之外运行数据库实例会将数据库从您的环境的生命周期中分离，并且让您可以从多个环境连接到同一数据库、将数据库互相交换或执行蓝/绿部署，而不会影响您的数据库。

Sections

- 先决条件
- 在 Amazon RDS 中启动数据库实例
- 启动 Elastic Beanstalk 环境
- 配置安全设置和环境属性
- 配置并部署您的应用程序
- 安装 Drupal
- 更新 Drupal 配置并删除访问限制
- 配置 Auto Scaling 组
- 清除
- 后续步骤

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

本教程中针对 Amazon Relational Database Service (Amazon RDS) 任务的过程假定您正在默认的 Amazon Virtual Private Cloud (Amazon VPC) 中启动资源。所有新账户在每个区域中均包含一个默认 VPC。如果您没有默认 VPC，则这些过程将会发生变化。有关 EC2-Classic 和自定义 VPC 平台的说明，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

此示例应用程序使用 Amazon EFS。它仅适用于支持 Amazon EFS 的 AWS 区域。要了解有关支持 AWS 区域的信息，请参阅中的 Amazon Elastic File System 终端节点和配额AWS 一般参考。

如果您的 Elastic Beanstalk 环境的平台使用 PHP 7.4 或更早版本，我们建议您在本教程中使用 Drupal 8.9.13 版本。对于安装有 PHP 8.0 或更高版本的平台，我们建议您使用 Drupal 9.1.5。

有关 Drupal 版本及其支持的 PHP 版本的更多信息，请参阅 Drupal 网站上的 PHP 要求。Drupal 推荐的核心版本已在以下网站上列出：https://www.drupal.org/project/drupal。

在 Amazon RDS 中启动数据库实例

要将外部数据库与在 Elastic Beanstalk 中运行的应用程序结合使用，请首先使用 Amazon RDS 启动数据库实例。当您使用 Amazon RDS 启动实例时，它完全独立于 Elastic Beanstalk 和您的 Elastic Beanstalk 环境，并且不会由 Elastic Beanstalk 终止或监控。

使用 Amazon RDS 控制台可启动多可用区 MySQL 数据库实例。选择多可用区部署可确保您的数据库将进行故障转移并在源数据库实例中断服务时继续可用。

在默认 VPC 中启动 RDS 数据库实例

1. 打开 RDS 控制台。

2. 在导航窗格中，选择 Databases（数据库）。
3. 选择创建数据库。
4. 选择 Standard Create（标准创建）。

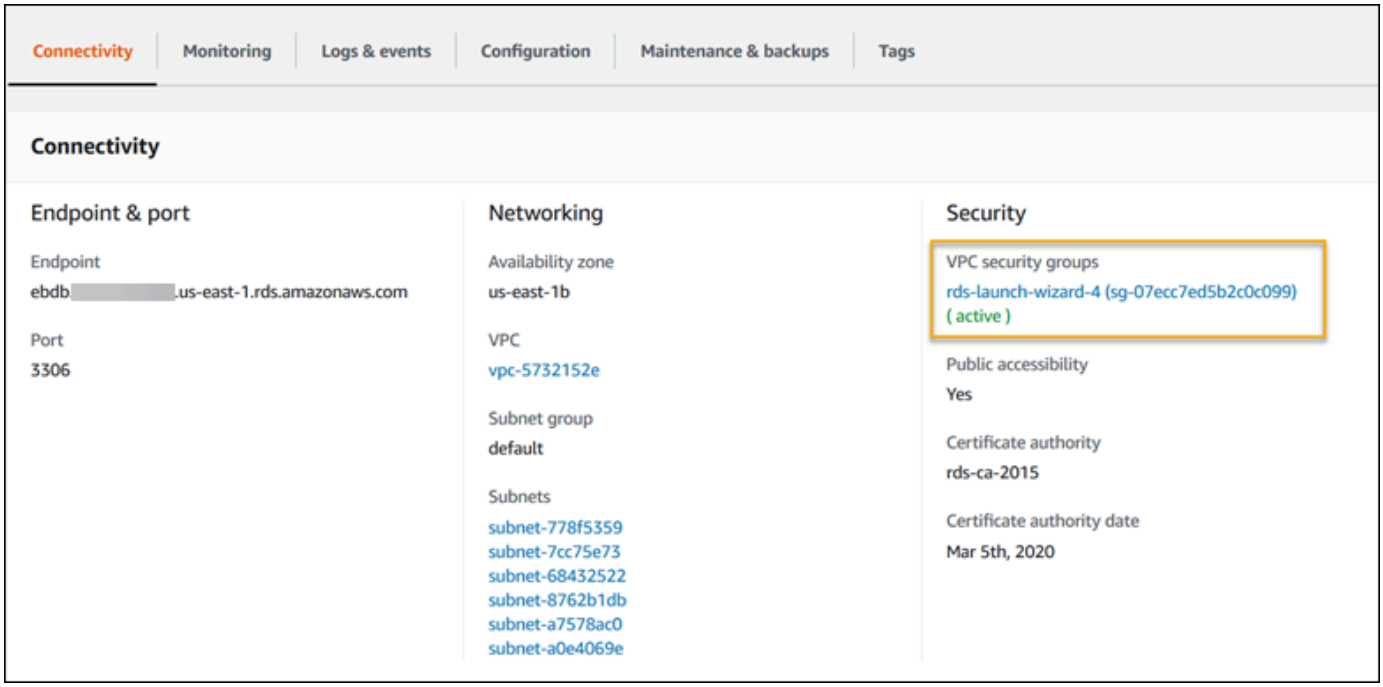
> **Important**
>
> 请勿选择 Easy Create（轻松创建）。如果您选择它，您将无法配置启动此 RDS 数据库所需的设置。

5. 在 Additional configuration（附加配置）下，对于 Initial database name（初始数据库名称），键入 **ebdb**。
6. 请查看默认设置并根据您的具体要求调整这些设置。请注意以下选项：
   - DB instance class（数据库实例类）– 选择对于您的工作负载具有适当的内存量和 CPU 能力的实例大小。
   - Multi-AZ deployment（多可用区部署）– 为了实现高可用性，请将其设置为 Create an Aurora Replica/Reader node in a different AZ（在不同的可用区中创建 Aurora 副本/读取器节点）。
   - Master username（主用户名）和 Master password（主密码）– 数据库用户名和密码。请记下这些设置，因为您以后将使用这些值。
7. 验证其余选项的默认设置，然后选择 Create database（创建数据库）。

接下来，修改附加到数据库实例的安全组以允许相应的端口上的入站流量。这与您稍后要附加到 Elastic Beanstalk 环境的安全组相同，因此您添加的规则将向同一安全组中的其他资源授予入口流量权限。

修改附加到 RDS 实例的安全组上的入站规则

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例的名称以查看其详细信息。
4. 在 Connectivity（连接）部分中，记下在该页上显示的 Subnets（子网）、Security groups（安全组）和 Endpoint（端点）。这样您稍后可以使用这些信息。
5. 在 Security（安全性）下面，您可以查看与数据库实例关联的安全组。打开链接以在 Amazon EC2 控制台中查看安全组。



6. 在安全组详细信息中，选择 Inbound（入站）选项卡。
7. 选择 Edit（编辑）。
8. 选择 Add Rule（添加规则）。
9. 对于 Type（类型），请选择应用程序使用的数据库引擎。
10. 对于 Source（源），键入 **sg-** 以查看可用安全组的列表。请选择 Elastic Beanstalk 环境中使用的与 Auto Scaling 组关联的安全组。这样，环境中的 Amazon EC2 实例就可以访问数据库。

11. 选择 Save（保存）。

创建一个数据库实例大约需要 10 分钟。同时，启动您的 Elastic Beanstalk 环境。

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 PHP 平台并接受默认设置和示例代码。在启动环境后，可以配置环境以连接到数据库，然后将 Drupal 代码部署到环境。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。

- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到 公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。您启动的 RDS 数据库实例位于您的环境之外，因此您需负责管理其生命周期。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

配置安全设置和环境属性

将数据库实例的安全组添加到正在运行的环境。此过程将使 Elastic Beanstalk 使用附加的其他安全组重新配置环境中的所有实例。

向环境添加安全组

- 请执行以下操作之一：
  - 使用 Elastic Beanstalk 控制台添加安全组

    a. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域

    b. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

    > **Note**
    >
    > 如果您有多个环境，请使用搜索栏筛选环境列表。

    c. 在导航窗格中，选择 Configuration（配置）。

d. 在 Instances（实例）配置类别中，选择 Edit（编辑）。
e. 在 EC2 security groups（EC2 安全组）下面，选择要附加到实例的安全组以及 Elastic Beanstalk 创建的实例安全组。
f. 要保存更改，请选择页面底部的 Apply（应用）。
g. 阅读警告，然后选择 Confirm（确认）。

- 要使用配置文件添加安全组，请使用 `securitygroup-addexisting.config` 示例文件。

接下来，使用环境属性将连接信息传递给环境。示例应用程序使用了默认的一组属性，这些属性与您在环境中预置数据库时 Elastic Beanstalk 配置的属性匹配。

为 Amazon RDS 数据库实例配置环境属性

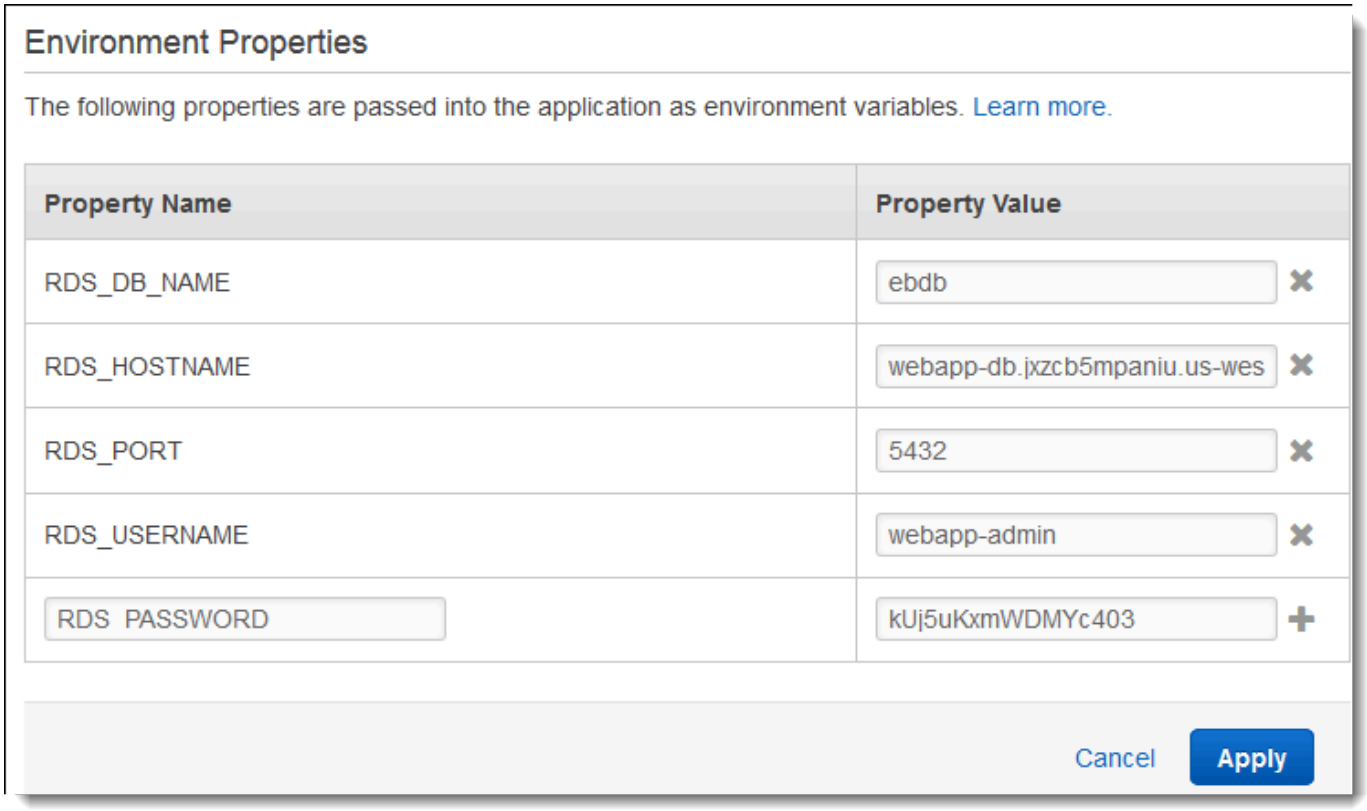
1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在环境属性部分中，定义应用程序读取的用于构建连接字符串的变量。为了实现与具有集成 RDS 数据库实例的环境的兼容，请使用以下名称和值。您可以在 RDS 控制台中找到除密码以外的所有值。

属性名称	描述	属性值
`RDS_HOSTNAME`	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
`RDS_PORT`	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。

属性名称	描述	属性值
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。



6. 要保存更改，请选择页面底部的 Apply（应用）。

安装 Drupal 后，您需要使用 SSH 连接到实例，以检索某些配置详细信息。将 SSH 密钥分配给环境的实例。

配置 SSH

1. 如果您之前未创建密钥对，请打开 Amazon EC2 控制台的密钥对页面，然后按照说明创建一个。
2. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
3. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

4. 在导航窗格中，选择 Configuration（配置）。
5. 在 Security（安全性）下，选择 Edit（编辑）。
6. 对于 EC2 密钥对，选择您的密钥对。
7. 要保存更改，请选择页面底部的 Apply（应用）。

配置并部署您的应用程序

要为 Elastic Beanstalk 创建 Drupal 项目，请下载 Drupal 源代码并将其与 aws-samples/ 存储库中的文件合并。eb-php-drupal GitHub

创建 Drupal 项目

1. 运行以下命令从 www.drupal.org/download 下载 Drupal。要了解有关下载的更多信息，请参阅 Drupal 网站。

   如果您的 Elastic Beanstalk 环境的平台使用 PHP 7.4 或更早版本，我们建议您在本教程中下载 Drupal 8.9.13 版本。您可以运行以下命令下载该版本。

   ```
   ~$ curl https://ftp.drupal.org/files/projects/drupal-8.9.13.tar.gz -o drupal.tar.gz
   ```

   如果您的平台使用 PHP 8.0 或更高版本，我们建议您下载 Drupal 9.1.5。您可以使用此命令下载它。

   ```
   ~$ curl https://ftp.drupal.org/files/projects/drupal-9.1.5.tar.gz -o drupal.tar.gz
   ```

有关 Drupal 版本及其支持的 PHP 版本的更多信息，请参阅 Drupal 官方文档中的 PHP 要求。Drupal 推荐的核心版本已在 Drupal 网站上列出。

2. 使用以下命令从示例存储库中下载配置文件：

```
~$ wget https://github.com/aws-samples/eb-php-drupal/releases/download/v1.1/eb-php-drupal-v1.zip
```

3. 提取 Drupal 并更改文件夹的名称。

   如果您已下载 Drupal 8.9.13：

```
~$ tar -xvf drupal.tar.gz
~$ mv drupal-8.9.13 drupal-beanstalk
~$ cd drupal-beanstalk
```

   如果您已下载 Drupal 9.1.5：

```
~$ tar -xvf drupal.tar.gz
~$ mv drupal-9.1.5 drupal-beanstalk
~$ cd drupal-beanstalk
```

4. 从 Drupal 安装中提取配置文件。

```
~/drupal-beanstalk$ unzip ../eb-php-drupal-v1.zip
 creating: .ebextensions/
 inflating: .ebextensions/dev.config
 inflating: .ebextensions/drupal.config
 inflating: .ebextensions/efs-create.config
 inflating: .ebextensions/efs-filesystem.template
 inflating: .ebextensions/efs-mount.config
 inflating: .ebextensions/loadbalancer-sg.config
 inflating: LICENSE
 inflating: README.md
 inflating: beanstalk-settings.php
```

验证您的 drupal-beanstalk 文件夹结构是否正确，如下所示。

```
drupal-beanstalk$ tree -aL 1
.
```

```
### autoload.php
### beanstalk-settings.php
### composer.json
### composer.lock
### core
### .csslintrc
### .ebextensions
### .ebextensions
### .editorconfig
### .eslintignore
### .eslintrc.json
### example.gitignore
### .gitattributes
### .htaccess
### .ht.router.php
### index.php
### LICENSE
### LICENSE.txt
### modules
### profiles
### README.md
### README.txt
### robots.txt
### sites
### themes
### update.php
### vendor
### web.config
```

项目存储库中的 `beanstalk-settings.php` 文件使用您在上一步中定义的环境变量来配置数据库连接。`.ebextensions` 文件夹包含用于在您的 Elastic Beanstalk 环境中创建其他资源的配置文件。

该配置文件要求您进行修改以使用账户。将文件中的占位符值替换为适当的 ID 并创建源代码包。

更新配置文件并创建源代码包。

1. 按如下方式修改配置文件。

   - `.ebextensions/dev.config` - 将对您的环境的访问限制为您的 IP 地址，以便在 Drupal 安装过程中保护环境。将靠近文件顶部的占位符 IP 地址替换为您的公有 IP 地址。
   - `.ebextensions/efs-create.config` - 在您的 VPC 中的每个可用区/子网中创建 EFS 文件系统和装载点。在 Amazon VPC 控制台中确定您的默认 VPC 和子网 ID。

2. 创建将文件包含在项目文件夹中的源代码包。以下命令将创建名为 drupal-beanstalk.zip 的源包。它将排除 vendor 文件夹中的文件，因为这些文件会占用大量空间并且对于将您的应用程序部署到 Elastic Beanstalk 不是必需的。

```
~/eb-drupal$ zip ../drupal-beanstalk.zip -r * .[^.]* -x "vendor/*"
```

将源包上传到 Elastic Beanstalk 以将 Drupal 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

安装 Drupal

完成 Drupal 安装

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 URL 以便在浏览器中打开您的站点。您将会重定向到 Drupal 安装向导，因为该站点还未配置。
4. 执行具有以下数据库设置的标准安装：

- 数据库名称 - Amazon RDS 控制台中显示的数据库名称。
- 数据库用户名和密码 - 您在创建数据库时输入的主用户名和主密码值。
- 高级选项 > 主机 - 在 Amazon RDS 控制台中显示的数据库实例的终端节点。

安装大约需要一分钟能完成。

更新 Drupal 配置并删除访问限制

Drupal 安装过程将在实例上的 `settings.php` 文件夹中创建一个名为 `sites/default` 的文件。您的源代码中需要此文件来避免在后续部署中重置您的站点，但该文件当前包含了您不希望提交到源的密钥。连接到应用程序实例以从设置文件中检索信息。

使用 SSH 连接到应用程序实例

1. 打开 Amazon EC2 控制台的实例页面。
2. 选择应用程序实例。它是以您的 Elastic Beanstalk 环境命名的实例。
3. 选择连接。
4. 按照以下说明将实例与 SSH 连接。该命令看上去类似以下内容。

```
$ ssh -i ~/.ssh/mykey ec2-user@ec2-00-55-33-222.us-west-2.compute.amazonaws.com
```

从最后一行设置文件中获取同步目录 ID。

```
[ec2-user ~]$ tail -n 1 /var/app/current/sites/default/settings.php
$config_directories['sync'] = 'sites/default/files/
config_4ccfX2sPQm79p1mk5IbUq9S_FokcENO4mxyC-L18-4g_xKj_7j9ydn31kDOYOgnzMu071Tvc4Q/
sync';
```

该文件还包含站点的当前哈希键，但您可以忽略当前值并使用您自己的值。

将同步目录路径和哈希键分配到环境属性。项目存储库中的自定义设置文件将读取这些属性以便在部署期间配置站点，并读取您之前设置的数据库连接属性。

Drupal 配置属性

- `SYNC_DIR` - 同步目录的路径。
- `HASH_SALT` - 符合环境属性要求的任何字符串值。

在 Elastic Beanstalk 控制台中配置环境属性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 向下滚动到环境属性。
6. 选择添加环境属性。
7. 输入属性名称和值对。
8. 如需添加更多变量，请重复步骤 6 和步骤 7。
9. 要保存更改，请选择页面底部的 Apply（应用）。

最后，示例项目包含一个创建安全组并将其分配给环境负载均衡器的配置文件 (`loadbalancer-sg.config`)，使用您在 `dev.config` 中配置的 IP 地址将端口 80 上的 HTTP 访问限制为来自您的网络的连接。否则，在您安装 Drupal 并配置您的管理员账户之前，外部方可能连接到您的站点。

更新 Drupal 的配置并删除访问限制

1. 从项目目录中删除 `.ebextensions/loadbalancer-sg.config` 文件。

```
~/drupal-beanstalk$ rm .ebextensions/loadbalancer-sg.config
```

2. 将自定义 `settings.php` 文件复制到站点文件夹中。

```
~/drupal-beanstalk$ cp beanstalk-settings.php sites/default/settings.php
```

3. 创建源包。

```
~/eb-drupal$ zip ../drupal-beanstalk-v2.zip -r * .[^.]* -x "vendor/*"
```

将源包上传到 Elastic Beanstalk 以将 Drupal 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

配置 Auto Scaling 组

最后，使用较高的最低实例计数配置您环境的 Auto Scaling 组。请始终至少运行两个实例，以防止您环境中的 Web 服务器发生单点故障，并支持您在不中断站点服务的情况下部署更改。

配置您的环境的 Auto Scaling 组以获得高可用性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在 Auto Scaling group（Auto Scaling 组）部分中，将 Min instances（最小实例数）设置为 **2**。
6. 要保存更改，请选择页面底部的 Apply（应用）。

为了支持多个实例中的内容上传，示例项目使用 Amazon Elastic File System 来创建共享文件系统。在站点上创建一个帖子，然后上传内容以将其存储在共享文件系统上。查看帖子并多次刷新页面，以命中两个实例并验证共享文件系统是否正常工作。

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

此外，您还可以终止在您的 Elastic Beanstalk 环境外部创建的数据库资源。当您终止 Amazon RDS 数据库实例时，可以拍摄快照并在以后将数据恢复到其他实例。

终止您的 RDS 数据库实例

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例。
4. 选择操作，然后选择删除。
5. 选择是否创建快照，然后选择删除。

后续步骤

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

示例应用程序使用配置文件配置 PHP 设置并在数据库中创建表 (如果表尚未存在)。您还可以使用配置文件在环境创建期间配置实例的安全组设置，以避免耗时的配置更新。请参阅使用配置文件 (.ebextensions) 进行高级环境自定义了解更多信息。

对于开发和测试，您可能希望使用 Elastic Beanstalk 的用于将托管数据库实例直接添加到环境的功能。有关在环境内设置数据库的说明，请参阅将数据库添加到 Elastic Beanstalk 环境。

如果您需要高性能数据库，请考虑使用 Amazon Aurora。Amazon Aurora 是与 MySQL 兼容的数据库引擎，能够以低成本提供商用数据库功能。要将您的应用程序连接到另一个数据库，请重复执行安全组配置步骤并更新与 RDS 相关的环境属性。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

向 PHP 应用程序环境中添加 Amazon RDS 数据库实例

您可以使用 Amazon Relational Database Service（Amazon RDS）数据库实例来存储由应用程序收集和修改的数据。数据库可以耦合到您的环境并由 Elastic Beanstalk 进行管理，也可以被创建为解耦数据库并由另一项服务进行外部管理。本主题提供使用 Elastic Beanstalk 控制台创建 Amazon RDS 的说明。数据库将耦合到您的环境并由 Elastic Beanstalk 进行管理。有关将 Amazon RDS 与 Elastic Beanstalk 集成的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

小节目录

向环境中添加数据库实例

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

添加一个数据库实例大约需要 10 分钟。环境更新完成后，您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息：

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

有关与 Elastic Beanstalk 环境耦合的数据库实例配置的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

下载驱动程序

要使用 PHP 数据对象 (PDO) 连接到数据库，请安装与您所选数据库引擎匹配的驱动程序。

- MySQL – PDO_MYSQL
- PostgreSQL – PDO_PGSQL
- Oracle – PDO_OCI
- SQL Server – PDO_SQLSRV

有关更多信息，请参阅http://php.net/manual/en/pdo.installation.php。

使用 PDO 或 MySQLi 连接到数据库

您可以使用 $_SERVER[`*VARIABLE*`] 从环境中读取连接信息。

对于 PDO，请根据主机、端口和名称创建一个数据源名称 (DSN)。使用数据库用户名和密码将 DSN 传递到 PDO 的构造函数。

Example 使用 PDO 连接到 RDS 数据库 - MySQL

```
<?php
$dbhost = $_SERVER['RDS_HOSTNAME'];
$dbport = $_SERVER['RDS_PORT'];
$dbname = $_SERVER['RDS_DB_NAME'];
$charset = 'utf8' ;

$dsn = "mysql:host={$dbhost};port={$dbport};dbname={$dbname};charset={$charset}";
$username = $_SERVER['RDS_USERNAME'];
$password = $_SERVER['RDS_PASSWORD'];

$pdo = new PDO($dsn, $username, $password);
?>
```

对于其他驱动程序，请将 mysql 替换为您的驱动程序的名称 - pgsql oci 或 sqlsrv。

对于 MySQLi，请将主机名、用户名、密码、数据库名称和端口传递到 mysqli 构造函数。

Example 使用 mysqli_connect() 连接到 RDS 数据库

```
$link = new mysqli($_SERVER['RDS_HOSTNAME'], $_SERVER['RDS_USERNAME'],
 $_SERVER['RDS_PASSWORD'], $_SERVER['RDS_DB_NAME'], $_SERVER['RDS_PORT']);
```

使用 Symfony 连接到数据库

对于 Symfony 版本 3.2 和更高版本，您可以使用 `%env(PROPERTY_NAME)%`，基于由 Elastic Beanstalk 设置的环境属性在配置文件中设置数据库参数。

Example app/config/parameters.yml

```
parameters:
    database_driver:   pdo_mysql
    database_host:     '%env(RDS_HOSTNAME)%'
    database_port:     '%env(RDS_PORT)%'
    database_name:     '%env(RDS_DB_NAME)%'
    database_user:     '%env(RDS_USERNAME)%'
    database_password: '%env(RDS_PASSWORD)%'
```

有关更多信息，请参阅外部参数 (Symfony 3.4)。

对于较早版本的 Symfony，环境变量仅在以 SYMFONY__ 开头时可供访问。这意味着 Elastic Beanstalk 定义的环境属性无法访问，并且您必须定义自己的环境属性以将连接信息传递给 Symfony。

要使用 Symfony 2 连接到数据库，请为每个参数创建一个环境属性。然后，使用 `%property.name%` 来访问配置文件中由 Symfony 转换的变量。例如，名为 SYMFONY__DATABASE__USER 的环境属性可作为 `database.user` 访问。

```
    database_user:     "%database.user%"
```

有关更多信息，请参阅外部参数 (Symfony 2.8)。

使用 Python

本部分提供使用 AWS Elastic Beanstalk 部署 Python 应用程序的教程和相关信息。

本章中的主题假设您对 Elastic Beanstalk 环境有所了解。如果您以前未使用过 Elastic Beanstalk，请尝试使用入门教程以了解基本知识。

主题

- 设置 Python 开发环境
- 使用 Elastic Beanstalk Python 平台

- 将 Flask 应用程序部署到 Elastic Beanstalk
- 将 Django 应用程序部署到 Elastic Beanstalk
- 向 Python 应用程序环境中添加 Amazon RDS 数据库实例
- Python 工具和资源

设置 Python 开发环境

设置 Python 开发环境以在本地测试应用程序，然后再将其部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

有关适用于所有语言的常见设置步骤和工具，请参阅配置您的开发计算机。

Sections

- 先决条件
- 使用虚拟环境
- 为 Elastic Beanstalk 配置 Python 项目

先决条件

对于所有您将使用 Elastic Beanstalk 部署的 Python 应用程序，这些先决条件都是常见的：

1. 与应用程序将使用的 Elastic Beanstalk Python 平台版本匹配的 Python 版本。
2. `pip` 实用程序与您的 Python 版本匹配。它用于安装并列出您的项目的依赖项，以便 Elastic Beanstalk 了解如何设置您的应用程序环境。
3. AWS Elastic Beanstalk 命令行界面 (EB CLI)。该程序包用于使用随 Elastic Beanstalk 部署所需的文件初始化您的应用程序。

4. 可工作的 ssh 安装。这用于在您需要检查或调试部署时连接运行的实例。
5. The virtualenv 程序包。该程序包用于创建开发和测试应用程序时使用的环境，这样您就无需安装应用程序所需的其他程序包，即可通过 Elastic Beanstalk 复制这一环境。使用以下命令安装此软件包：

```
$ pip install virtualenv
```

有关安装 Python、pip 和 EB CLI 的说明，请参阅安装 EB CLI。

使用虚拟环境

在安装先决条件软件后，使用 virtualenv 设置虚拟环境以安装您的应用程序的依赖项。通过使用虚拟环境，您可以确切地识别哪些程序包是您的应用程序所必需的，以便在运行应用程序的 EC2 实例上安装所需的数据包。

设置虚拟环境

1. 打开命令行窗口并键入：

   ```
   $ virtualenv /tmp/eb_python_app
   ```

   使用对您的应用程序有意义的名称替换 *eb_python_app*（使用您的应用程序的名称是一个不错的方法）。virtualenv 命令在指定目录中为您创建一个虚拟环境，并打印其操作的结果：

   ```
   Running virtualenv with interpreter /usr/bin/python
   New python executable in /tmp/eb_python_app/bin/python3.7
   Also creating executable in /tmp/eb_python_app/bin/python
   Installing setuptools, pip...done.
   ```

2. 一旦您的虚拟环境准备就绪，请通过运行位于环境 activate 目录中的 bin 脚本启动它。例如，要启动在上一步骤中创建的 eb_python_app 环境，您需要键入：

   ```
   $ source /tmp/eb_python_app/bin/activate
   ```

   虚拟环境在各个命令提示符的开头输出其名称 (例如：(eb_python_app))，提醒您处于虚拟 Python 环境中。
3. 要停止使用虚拟环境并返回系统的默认 Python 解释器及其所有已安装的库，请运行 deactivate 命令。

```
(eb_python_app) $ deactivate
```

> **Note**
>
> 创建之后，您可以随时通过重新运行虚拟环境的 `activate` 脚本来重新启动它。

为 Elastic Beanstalk 配置 Python 项目

您可以使用 Elastic Beanstalk CLI 来准备 Python 应用程序，以便与 Elastic Beanstalk 一起部署。

将 Python 应用程序配置为随 Elastic Beanstalk 一起部署

1. 从您的虚拟环境中，返回项目目录树的顶层 (`python_eb_app`)，然后键入：

   ```
   pip freeze >requirements.txt
   ```

   此命令将已安装到您虚拟环境中的程序包的名称和版本复制到 `requirements.txt`。例如，如果虚拟系统上安装了 PyYAML 程序包版本 3.11，则文件将包含以下行：

   ```
   PyYAML==3.11
   ```

   这使得 Elastic Beanstalk 可以使用您在开发和测试应用程序时使用的相同程序包和相同版本来复制应用程序的 Python 环境。
2. 使用 eb init 命令配置 EB CLI 存储库。按照提示选择区域、平台和其他选项。有关详细说明，请参阅 使用 EB CLI 管理 Elastic Beanstalk 环境。

默认情况下，Elastic Beanstalk 会查找名为 `application.py` 的文件来启动您的应用程序。如果这不在您创建的 Python 项目中，则必须对您的应用程序环境做出一些调整。您还需要设置环境变量，以便可以加载您的应用程序模块。参阅 使用 Elastic Beanstalk Python 平台 了解更多信息。

使用 Elastic Beanstalk Python 平台

AWS Elastic Beanstalk Python 平台是一组平台版本，适用于可在代理服务器后方使用 WSGI 运行的 Python Web 应用程序。每个平台分支均对应于 Python 的一个版本，例如 Python 3.8。

从 Amazon Linux 2 平台分支开始，Elastic Beanstalk 将 Gunicorn 作为默认 WSGI 服务器。

您可以将 `Procfile` 添加到源包，用于指定和配置应用程序的 WSGI 服务器。有关详细信息，请参阅 the section called “Procfile”。

您可以使用通过 Pipenv 创建的 `Pipfile` 和 `Pipfile.lock` 文件，指定 Python 包依赖项和其他要求。有关指定依赖项的详细信息，请参阅the section called “指定依赖项”。

Elastic Beanstalk 提供了配置选项，可供您用于自定义在 Elastic Beanstalk 环境中的 EC2 实例上运行的软件。您可配置应用程序所需的环境变量，启用日志轮换至 Amazon S3，并将应用程序源中包含静态文件的文件夹映射至代理服务器所提供的路径。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

对于 `pip` 提供的 Python 包，您可以在应用程序源代码的根目录中包含要求文件。Elastic Beanstalk 将在部署期间安装要求文件中指定的所有依赖项程序包。有关详细信息，请参阅 the section called “指定依赖项”。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅the section called “扩展 Linux 平台”。

配置 Python 环境

Python 平台设置允许您微调 Amazon EC2 实例的行为。您可以使用 Elastic Beanstalk 控制台编辑 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

使用 Elastic Beanstalk 控制台配置 Python 进程设置，启用 AWS X-Ray，启用到 Amazon S3 的日志轮换，并配置应用程序可以从环境中读取的变量。

在 Elastic Beanstalk 控制台中配置 Python 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

Python 设置

- Proxy server（代理服务器）– 要在环境实例上使用的代理服务器。默认情况下，使用 nginx。
- WSGI Path（WSGI 路径）– 主应用程序文件的名称或路径。例如，`application.py` 或 `django/wsgi.py`。
- NumProcesses - 要在每个应用程序实例上运行的流程的数量。
- NumThreads - 要在每个流程中运行的线程的数量。

AWS X-Ray 设置

- X-Ray daemon（X-Ray 守护进程）– 运行 AWS X-Ray 守护进程可处理来自 AWS X-Ray SDK for Python 的跟踪数据。

日志选项

“日志选项”部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

默认情况下，Python 环境中的代理服务器提供 `static` 路径下名为 `/static` 的文件夹中的任何文件。例如，如果您的应用程序源代码在名为 `logo.png` 的文件夹中包含一个名为 `static` 的文件，则代理服务器将通过 *`subdomain`*`.elasticbeanstalk.com/static/logo.png` 将其提供给用户。您可以配置额外的映射，如本节中所述。

环境属性

您可使用环境属性来为应用程序提供信息，配置环境变量。例如，您可创建名为 `CONNECTION_STRING` 的环境属性，指定应用程序可用于连接数据库的连接字符串。

在 Elastic Beanstalk 上运行的 Python 环境中，可以使用 Python 的 `os.environ` 字典访问这些值。有关更多信息，请参阅 http://docs.python.org/library/os.html。

您可使用类似以下内容的代码，用于访问各种密钥和参数：

```
import os
endpoint = os.environ['API_ENDPOINT']
```

环境属性也可为框架提供信息。例如，您可创建名为 `DJANGO_SETTINGS_MODULE` 的属性，以配置 Django 为使用特定设置模块。根据环境的不同，值可以是 `development.settings`、`production.settings` 等等。

参阅 环境属性和其他软件设置 了解更多信息。

Python 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

Python 平台在 `aws:elasticbeanstalk:environment:proxy`、`aws:elasticbeanstalk:environment:proxy:st` 和 `aws:elasticbeanstalk:container:python` 命名空间中定义选项。

以下示例配置文件指定配置选项设置以创建一个名为 `DJANGO_SETTINGS_MODULE` 的环境属性，选择 Apache 代理服务器，指定两个静态文件选项（用于将名为 `statichtml` 的目录映射至路径 `/html` 和将名为 `staticimages` 的目录映射到路径 `/images`）以及指定

aws:elasticbeanstalk:container:python 命名空间中的附加设置。此命名空间包含让您在源代码中指定 WSGI 脚本位置的选项，以及运行 WSGI 的线程和进程数量。

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    DJANGO_SETTINGS_MODULE: production.settings
  aws:elasticbeanstalk:environment:proxy:
    ProxyServer: apache
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /html: statichtml
    /images: staticimages
  aws:elasticbeanstalk:container:python:
    WSGIPath: ebdjango.wsgi:application
    NumProcesses: 3
    NumThreads: 20
```

注意

- 如果您使用的是 Amazon Linux AMI Python 平台版本（在 Amazon Linux 2 之前），请将 WSGIPath 的值替换为 ebdjango/wsgi.py。示例中的值适用于 Gunicorn WSGI 服务器，该服务器在 Amazon Linux AMI 平台版本上不受支持。
- 此外，这些较旧的平台版本使用不同的命名空间来配置静态文件 - aws:elasticbeanstalk:container:python:staticfiles。它与标准静态文件命名空间具有相同的选项名称和语义。

配置文件还支持多个密钥，以便进一步在环境实例上修改软件。此示例使用程序包密钥，以使用 yum 和容器命令安装 Memcached，从而在部署期间运行配置服务器的命令。

```
packages:
  yum:
    libmemcached-devel: '0.31'

container_commands:
  collectstatic:
    command: "django-admin.py collectstatic --noinput"
  01syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true
  02migrate:
```

```
    command: "django-admin.py migrate"
    leader_only: true
  03wsgipass:
    command: 'echo "WSGIPassAuthorization On" >> ../wsgi.conf'
  99customize:
    command: "scripts/customize.sh"
```

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

使用 Procfile 配置 WSGI 服务器

您可以将 `Procfile` 添加到源包，用于指定和配置应用程序的 WSGI 服务器。下面的示例使用 `Procfile` 将 UWSGI 指定为服务器并进行配置。

Example Procfile

```
web: uwsgi --http :8000 --wsgi-file application.py --master --processes 4 --threads 2
```

下面的示例使用 `Procfile` 配置 Gunicorn（即默认 WSGI 服务器）。

Example Procfile

```
web: gunicorn --bind :8000 --workers 3 --threads 2 project.wsgi:application
```

> **注意**
>
> - 如果您配置 Gunicorn 之外的任何 WSGI 服务器，请确保还将该服务器指定为应用程序的依赖项，以便将安装在环境实例上。有关依赖项规范的详细信息，请参阅the section called “指定依赖项”。
> - WSGI 服务器的默认端口为 8000。如果您在 `Procfile` 命令中指定了其他端口号，请将 PORT 环境属性也设置为该端口号。

使用 `Procfile` 时，它会覆盖您使用配置文件设置的 `aws:elasticbeanstalk:container:python` 命名空间选项。

有关 `Procfile` 用法的详细信息，请展开 the section called “扩展 Linux 平台” 中的构建文件和 Procfile 部分。

使用要求文件指定依赖项

典型的 Python 应用程序要依靠其他第三方的 Python 程序包。使用 Elastic Beanstalk Python 平台，您可以通过几种方法来指定应用程序所依赖的 Python 包。

使用 **pip** 和 **requirements.txt**

用于安装 Python 包的标准工具为 `pip`。该工具有一个功能，可让您在一个要求文件中指定所有必需的包（及其版本）。有关需求文件的更多信息，请参阅 pip 文档网站上的需求文件格式。

创建名为 `requirements.txt` 的文件，并将该文件放在源包的顶级目录中。以下是 Django 的示例 `requirements.txt` 文件。

```
Django==2.2
mysqlclient==2.0.3
```

在开发环境中，您可以使用 `pip freeze` 命令以生成要求文件。

```
~/my-app$ pip freeze > requirements.txt
```

为确保要求文件仅包含应用程序实际使用的程序包，请使用仅安装了这些程序包的虚拟环境。在虚拟环境外，`pip freeze` 的输出将包括在开发计算机上安装的所有 `pip` 程序包，包括操作系统随附的程序包。

> **Note**
>
> 在 Amazon Linux AMI Python 平台版本上，Elastic Beanstalk 本机不支持 Pipenv 或 Pipfiles。如果您使用 Pipenv 管理您应用程序的依赖性，请运行以下命令以生成一个 `requirements.txt` 文件。
>
> ```
> ~/my-app$ pipenv lock -r > requirements.txt
> ```
>
> 要了解更多信息，请参阅 Pipenv 文档中的生成 requirements.txt。

使用 Pipenv 和 **Pipfile**

Pipenv 是一个现代化的 Python 打包工具。该工具将软件包安装与创建和管理应用程序的依赖项文件和 virtualenv 结合起来。有关更多信息，请参阅 Pipenv：适合人工使用的 Python 开发工作流。

Pipenv 维护两个文件：

- `Pipfile` – 此文件包含各种类型的依赖项和要求。
- `Pipfile.lock` – 此文件包含启用确定性构建的版本快照。

您可以在开发环境中创建这些文件，并将这些文件包含在部署到 Elastic Beanstalk 的源包的顶级目录中。有关这两个文件的更多信息，请参阅示例 Pipfile 和 Pipfile.lock。

下面的示例使用 Pipenv 来安装 Django 和 Django REST 框架。这些命令会创建 `Pipfile` 和 `Pipfile.lock` 文件。

```
~/my-app$ pipenv install django
~/my-app$ pipenv install djangorestframework
```

优先顺序

如果您包含此主题中描述的多个要求文件，则 Elastic Beanstalk 只使用其中一个文件。下面的列表按降序顺序显示优先级。

1. `requirements.txt`
2. `Pipfile.lock`
3. `Pipfile`

> **Note**
>
> 从 2023 年 3 月 7 日的 Amazon Linux 2 平台版本开始，如果您提供这些文件中的多个，则 Elastic Beanstalk 将发布一条控制台消息，说明部署期间使用了哪个依赖项文件。

以下步骤描述了 Elastic Beanstalk 在部署实例时安装依赖项所遵循的逻辑。

- 如果有 `requirements.txt` 文件，则使用命令 `pip install -r requirements.txt`。
- 从 2023 年 3 月 7 日的 Amazon Linux 2 平台版本开始，如果没有 `requirements.txt` 文件但有 `Pipfile.lock`，则将使用命令 `pipenv sync`。在该版本之前，使用 `pipenv install --ignore-pipfile`。

- 如果既没有 `requirements.txt` 文件也没有 `Pipfile.lock`，但有 `Pipfile`，则使用命令 `pipenv install --skip-lock`。
- 如果找不到三个需求文件中的任何一个，则将不会安装任何应用程序依赖项。

将 Flask 应用程序部署到 Elastic Beanstalk

Flask 是一种适用于 Python 的开源 Web 应用程序框架。本教程将引导您完成生成 Flask 应用程序并将其部署到 AWS Elastic Beanstalk 环境的过程。

在本教程中，您将执行以下操作：

- 通过 Flask 设置 Python 虚拟环境
- 创建 Flask 应用程序
- 使用 EB CLI 部署站点
- 清除

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的 Ubuntu 和 Bash 版本。

Flask 需要 Python 3.7 或更高版本。在本教程中，我们使用 Python 3.7 和相应的 Elastic Beanstalk 平台版本。按照设置 Python 开发环境上的说明操作来安装 Python。

Flask 框架将作为此教程的一部分安装。

本教程还使用 Elastic Beanstalk 命令行界面 (EB CLI)。有关安装和配置 EB CLI 的详细信息，请参阅安装 EB CLI 和 配置 EB CLI。

通过 Flask 设置 Python 虚拟环境

为应用程序创建项目目录和虚拟环境并安装 Flask。

设置项目环境

1. 创建项目目录。

```
~$ mkdir eb-flask
~$ cd eb-flask
```

2. 创建和激活名为 `virt` 的虚拟环境：

```
~/eb-flask$ virtualenv virt
~$ source virt/bin/activate
(virt) ~/eb-flask$
```

   您将看到您的命令提示符前面带有 `(virt)`，表明您在虚拟环境中。在本教程的其余部分中使用虚拟环境。

3. 使用 `pip install` 安装 Flask：

```
(virt)~/eb-flask$ pip install flask==2.0.3
```

4. 使用 `pip freeze` 查看已安装的库：

```
(virt)~/eb-flask$ pip freeze
click==8.1.1
Flask==2.0.3
itsdangerous==2.1.2
Jinja2==3.1.1
MarkupSafe==2.1.1
Werkzeug==2.1.0
```

   此命令列出虚拟环境中已安装的所有程序包。由于您在虚拟环境中，因此不会显示全局安装的程序包，例如 EB CLI。

5. 将来自 `pip freeze` 的输出保存到名为 `requirements.txt` 的文件中。

```
(virt)~/eb-flask$ pip freeze > requirements.txt
```

在部署期间，此文件将指示 Elastic Beanstalk 安装库。有关更多信息，请参阅 使用要求文件指定依赖项。

创建 Flask 应用程序

接下来，创建一个您将使用 Elastic Beanstalk 部署的应用程序。我们会创建一个“Hello World”RESTful Web 服务。

在此目录中创建名为 `application.py` 的新文本文件，包含以下内容：

Example **`~/eb-flask/application.py`**

```
from flask import Flask

# print a nice greeting.
def say_hello(username = "World"):
    return '<p>Hello %s!</p>\n' % username

# some bits of text for the page.
header_text = '''
    <html>\n<head> <title>EB Flask Test</title> </head>\n<body>'''
instructions = '''
    <p><em>Hint</em>: This is a RESTful web service! Append a username
    to the URL (for example: <code>/Thelonious</code>) to say hello to
    someone specific.</p>\n'''
home_link = '<p><a href="/">Back</a></p>\n'
footer_text = '</body>\n</html>'

# EB looks for an 'application' callable by default.
application = Flask(__name__)

# add a rule for the index page.
application.add_url_rule('/', 'index', (lambda: header_text +
    say_hello() + instructions + footer_text))

# add a rule when the page is accessed with a name appended to the site
# URL.
application.add_url_rule('/<username>', 'hello', (lambda username:
    header_text + say_hello(username) + home_link + footer_text))

# run the app.
if __name__ == "__main__":
```

```
    # Setting debug to True enables debug output. This line should be
    # removed before deploying a production app.
    application.debug = True
    application.run()
```

本示例输出根据访问服务所使用的路径而变化的自定义问候语。

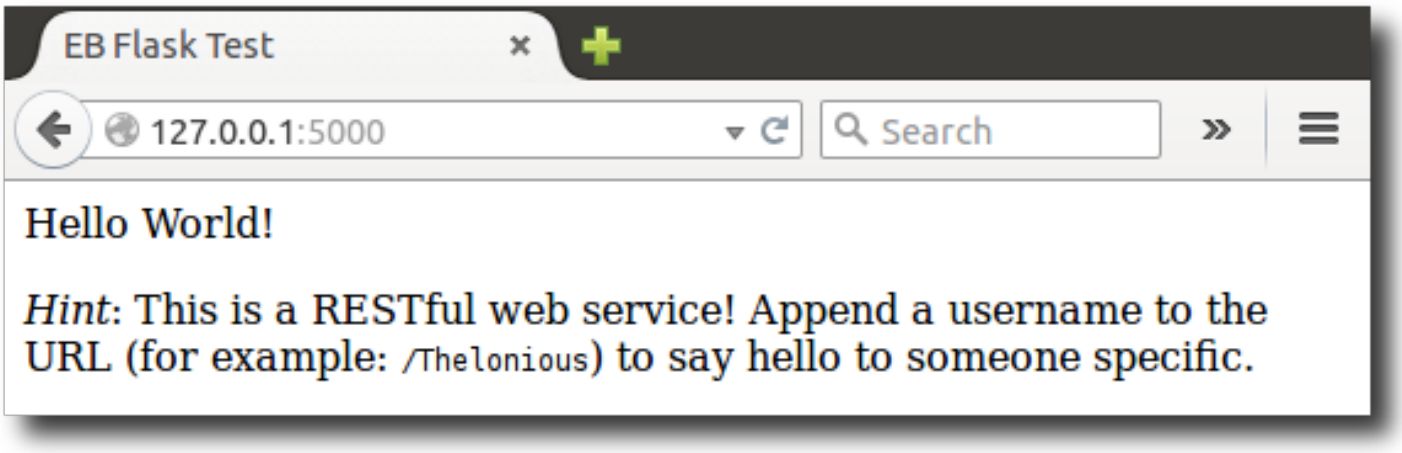
> **Note**
>
> 通过在运行应用程序之前添加 `application.debug = True`，可以启用调试输出以防出现问题。这对于开发工作来说是一个很好的做法，但是您应该在生产代码中删除调试语句，因为调试输出会暴露应用程序的内部机制。

使用 `application.py` 作为文件名并提供可调用的 `application` 对象（在本示例中为 Flask 对象）可允许 Elastic Beanstalk 轻松地找到您的应用程序代码。

将 `application.py` 与 Python 一起运行：

```
(virt) ~/eb-flask$ python application.py
 * Serving Flask app "application" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 313-155-123
```

在您的 Web 浏览器中打开 `http://127.0.0.1:5000/`。您应该看到应用程序正在运行，并且会显示索引页面：



检查服务器日志，查看请求的输出。您可键入 Ctrl+C，停止 Web 服务器并返回到虚拟环境。

如果您看到的是调试输出，请修复错误并确保应用程序正在本地运行，然后再为 Elastic Beanstalk 配置应用程序。

使用 EB CLI 部署站点

您已添加在 Elastic Beanstalk 上部署应用程序所需的全部内容。您的项目目录现在应该如下所示：

```
~/eb-flask/
|-- virt
|-- application.py
`-- requirements.txt
```

不过，不需要 `virt` 文件夹即可使应用程序在 Elastic Beanstalk 上运行。在部署时，Elastic Beanstalk 会在服务器实例上创建新的虚拟环境，并安装 `requirements.txt` 中列出的库。要最大程度地减小部署期间上传的源包的大小，请添加 .ebignore 文件，此文件告知 EB CLI 忽略 `virt` 文件夹。

Example ~/eb-flask/.ebignore

```
virt
```

下一步，您将创建应用程序环境并使用 Elastic Beanstalk 部署已配置的应用程序。

创建环境和部署 Flask 应用程序

1. 使用 eb init 命令，初始化 EB CLI 存储库：

```
~/eb-flask$ eb init -p python-3.7 flask-tutorial --region us-east-2
```

```
Application flask-tutorial has been created.
```

此命令创建名为 flask-tutorial 的新应用程序，并配置本地存储库，以最新的 Python 3.7 平台版本创建环境。

2. (可选) 再次运行 eb init，以配置默认密钥对，以便连接到使用 SSH 运行应用程序的 EC2 实例：

```
~/eb-flask$ eb init
Do you want to set up SSH for your instances?
(y/n): y
Select a keypair.
1) my-keypair
2) [ Create new KeyPair ]
```

如果您已有密钥对，请选择一个密钥对，或按提示创建新密钥对。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。

3. 使用 eb create 创建环境并将应用程序部署到此环境中：

```
~/eb-flask$ eb create flask-env
```

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。

- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

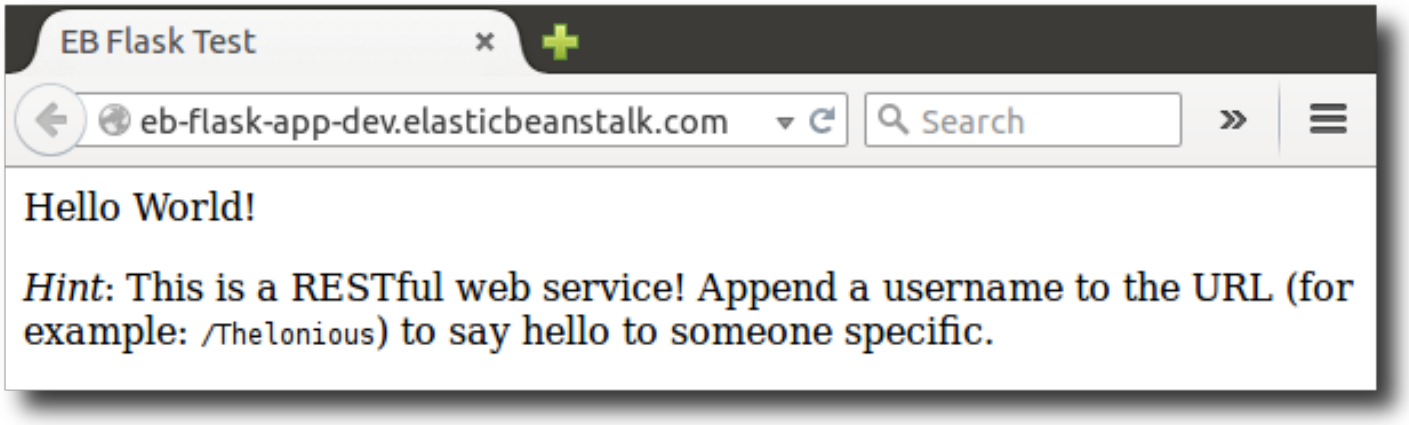
所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

当环境创建过程完成时，请使用 eb open 打开网站：

```
~/eb-flask$ eb open
```

这将使用为应用程序创建的域名打开一个浏览器窗口。您应该看到在本地创建和测试的相同 Flask 网站。



如果您没有看到应用程序运行，或者出现错误消息，请查看排查部署问题以获取有关如何确定错误原因的帮助。

如果您确实 看到应用程序在运行，那么恭喜，您已使用 Elastic Beanstalk 部署了第一个 Flask 应用程序！

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

或者，使用 EB CLI：

```
~/eb-flask$ eb terminate flask-env
```

后续步骤

有关 Flask 的更多信息，请访问 flask.pocoo.org。

如果您希望尝试其他 Python Web 框架，请参阅将 Django 应用程序部署到 Elastic Beanstalk。

将 Django 应用程序部署到 Elastic Beanstalk

本教程演练如何将默认的自动生成的 Django 网站部署到运行 Python 的 AWS Elastic Beanstalk 环境。本教程向您展示如何通过使用 Elastic Beanstalk 环境在云中托管 Python Web 应用程序。

在本教程中，您将执行以下操作：

- 设置 Python 虚拟环境和安装 Django
- 创建 Django 项目
- 为 Elastic Beanstalk 配置您的 Django 应用程序
- 使用 EB CLI 部署您的网站
- 更新应用程序
- 清除

先决条件

要使用任意 AWS 服务（包括 Elastic Beanstalk），您需要具有 AWS 账户和凭证。要了解更多信息并注册，请访问 https://aws.amazon.com/。

要完成此教程，您应当安装所有面向 Python 的常见先决条件软件，包括以下程序包：

- Python 3.7 或更高版本
- `pip`
- `virtualenv`
- `awsebcli`

Django 框架将作为此教程的一部分安装。

> **Note**
>
> 使用 EB CLI 创建环境需要服务角色。您可以通过在 Elastic Beanstalk 控制台中创建环境来创建服务角色。如果您没有服务角色，EB CLI 会尝试在您运行 `eb create` 时创建一个。

设置 Python 虚拟环境和安装 Django

使用 `virtualenv` 创建虚拟环境，并用它来安装 Django 及其依赖项。通过使用虚拟环境，您可以确切地了解哪些程序包是您的应用程序所必需的，以便在运行应用程序的 Amazon EC2 实例上安装所需的数据包。

以下步骤演示必须为基于 Unix 的系统和 Windows 输入的命令，这些命令显示在单独的选项卡上。

设置虚拟环境

1. 创建名为 `eb-virt` 的虚拟环境。

   Unix-based systems

   ```
   ~$ virtualenv ~/eb-virt
   ```

   Windows

   ```
   C:\> virtualenv %HOMEPATH%\eb-virt
   ```

2. 激活虚拟环境。

   Unix-based systems

   ```
   ~$ source ~/eb-virt/bin/activate
   (eb-virt) ~$
   ```

   Windows

   ```
   C:\>%HOMEPATH%\eb-virt\Scripts\activate
   (eb-virt) C:\>
   ```

   您将看到您的命令提示符前面带有 `(eb-virt)`，表明您在虚拟环境中。

> **Note**
>
> 这些指令的剩余部分在您的主目录 ~$ 中显示 Linux 命令提示符。在 Windows 上，这是 C:\Users*USERNAME*>，其中 *USERNAME* 是 Windows 登录名称。

3. 使用 pip 安装 Django。

```
(eb-virt)~$ pip install django==2.2
```

> **Note**
>
> 您安装的 Django 版本必须与您在 Elastic Beanstalk Python 配置中选择用于部署应用程序的 Python 版本兼容。有关部署的更多信息，请参阅本主题中的???。
> 有关当前 Python 平台版本的更多信息，请参阅《AWS Elastic Beanstalk 平台》文档中的 Python。
> 有关 Django 与 Python 的版本兼容性问题，请参阅我可以将哪个 Python 版本与 Django 一起使用？

4. 要验证已安装 Django，请输入以下命令。

```
(eb-virt)~$ pip freeze
Django==2.2
...
```

此命令列出虚拟环境中已安装的所有程序包。稍后，您使用此命令的输出来配置项目，以便配合 Elastic Beanstalk 使用。

创建 Django 项目

现在您已准备好使用虚拟环境来创建 Django 项目并在计算机上运行。

> **Note**
>
> 本教程使用 SQLite，这是包含在 Python 中的一个数据库引擎。使用您的项目文件部署数据库。对于生产环境，我们建议您使用 Amazon Relational Database Service (Amazon RDS)，

并且将它与您的环境分隔开。有关更多信息，请参阅 向 Python 应用程序环境中添加 Amazon RDS 数据库实例。

生成 Django 应用程序

1. 激活虚拟环境。

   Unix-based systems

   ```
   ~$ source ~/eb-virt/bin/activate
   (eb-virt) ~$
   ```

   Windows

   ```
   C:\>%HOMEPATH%\eb-virt\Scripts\activate
   (eb-virt) C:\>
   ```

   您将看到您的命令提示符带有 (eb-virt) 前缀，表明您在虚拟环境中。

   > **Note**
   >
   > 这些指令的剩余部分显示在您的主目录和 Linux 主目录 ~$ 中的 Linux 命令提示符 ~/。在 Windows 上，这是 C:\Users*USERNAME*>，其中 *USERNAME* 是您的 Windows 登录名称。

2. 使用 django-admin startproject 命令创建名为 ebdjango 的 Django 项目。

   ```
   (eb-virt)~$ django-admin startproject ebdjango
   ```

   该命令创建名为 ebdjango 的标准 Django 站点，包含以下目录结构。

   ```
   ~/ebdjango
     |-- ebdjango
     |   |-- __init__.py
     |   |-- settings.py
     |   |-- urls.py
     |   `-- wsgi.py
   ```

```
  `-- manage.py
```

3. 使用 `manage.py runserver` 在本地运行 Django 站点。

```
(eb-virt) ~$ cd ebdjango
```

```
(eb-virt) ~/ebdjango$ python manage.py runserver
```

4. 在 Web 浏览器中，打开 `http://127.0.0.1:8000/` 以查看站点。
5. 检查服务器日志，查看请求的输出。要停止 Web 服务器并返回到虚拟环境，请按 Ctrl+C。

```
Django version 2.2, using settings 'ebdjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[07/Sep/2018 20:14:09] "GET / HTTP/1.1" 200 16348
Ctrl+C
```

为 Elastic Beanstalk 配置您的 Django 应用程序

本地计算机已有一个 Django 支持的网站，因此，您可将其配置为配合 Elastic Beanstalk 部署。

默认情况下，Elastic Beanstalk 会查找名为 `application.py` 的文件来启动您的应用程序。由于这不在您创建的 Django 项目中，因此您必须对您应用程序的环境做出一些调整。您还必须设置环境变量，以便可以加载您的应用程序的模块。

为 Elastic Beanstalk 配置您的站点

1. 激活虚拟环境。

   Unix-based systems

   ```
   ~/ebdjango$ source ~/eb-virt/bin/activate
   ```

   Windows

   ```
   C:\Users\USERNAME\ebdjango>%HOMEPATH%\eb-virt\Scripts\activate
   ```

2. 运行 `pip freeze`，然后将输出保存到名为 `requirements.txt` 的文件。

```
(eb-virt) ~/ebdjango$ pip freeze > requirements.txt
```

Elastic Beanstalk 使用 `requirements.txt` 来确定哪些程序包应安装在运行应用程序的 EC2 实例上。

3. 创建名为 `.ebextensions` 的目录。

```
(eb-virt) ~/ebdjango$ mkdir .ebextensions
```

4. 在 `.ebextensions` 目录中，添加名为 `django.config` 的配置文件，包含以下文本：

Example ~/ebdjango/.ebextensions/django.config

```
option_settings:
  aws:elasticbeanstalk:container:python:
    WSGIPath: ebdjango.wsgi:application
```

设置 `WSGIPath` 指定 Elastic Beanstalk 用于启动应用程序的 WSGI 脚本位置。

> **Note**
>
> 如果您使用的是 Amazon Linux AMI Python 平台版本（在 Amazon Linux 2 之前），请将 `WSGIPath` 的值替换为 `ebdjango/wsgi.py`。示例中的值适用于 Gunicorn WSGI 服务器，该服务器在 Amazon Linux AMI 平台版本上不受支持。

5. 使用 `deactivate` 命令停用虚拟环境。

```
(eb-virt) ~/ebdjango$ deactivate
```

每当您需要为应用程序添加程序包或本地运行应用程序时，重新激活虚拟环境。

使用 EB CLI 部署您的网站

您已添加在 Elastic Beanstalk 上部署应用程序所需的全部内容。您的项目目录现在应该如下所示。

```
~/ebdjango/
|-- .ebextensions
|   `-- django.config
|-- ebdjango
```

```
|   |-- __init__.py
|   |-- settings.py
|   |-- urls.py
|   `-- wsgi.py
|-- db.sqlite3
|-- manage.py
`-- requirements.txt
```

下一步，您将创建应用程序环境并使用 Elastic Beanstalk 部署已配置的应用程序。

完成部署后，您应立即编辑 Django 的配置以将 Elastic Beanstalk 分配给您的应用程序的域名添加到 Django 的 ALLOWED_HOSTS。然后，您将重新部署您的应用程序。这是一个 Django 安全要求，旨在防止 HTTP Host 标头攻击。有关更多信息，请参阅主机标头验证。

创建环境并部署 Django 应用程序

> **Note**
>
> 本教程将 EB CLI 作为部署机制，但您也可以使用 Elastic Beanstalk控制台部署包含项目内容的 .zip 文件。

1. 使用 eb init 命令，初始化 EB CLI 存储库。

   ```
   ~/ebdjango$ eb init -p python-3.7 django-tutorial
   Application django-tutorial has been created.
   ```

   此命令会创建一个名为 django-tutorial 的应用程序。它还配置本地存储库，以最新的 Python 3.7 平台版本创建环境。

2. （可选）再次运行 eb init 以配置默认密钥对，以便使用 SSH 连接到运行您的应用程序的 EC2 实例。

   ```
   ~/ebdjango$ eb init
   Do you want to set up SSH for your instances?
   (y/n): y
   Select a keypair.
   1) my-keypair
   2) [ Create new KeyPair ]
   ```

如果您已有密钥对，请选择一个，或按提示创建一个。如果您没有看到提示或需要以后更改设置，请运行 eb init -i。

3. 创建环境并使用 eb create 将应用程序部署到此环境中。

```
~/ebdjango$ eb create django-env
```

> **Note**
>
> 如果您看到“service role required”错误消息，请以交互方式运行 `eb create` (不指定环境名称)，EB CLI 会为您创建角色。

此命令创建名为 `django-env` 的负载均衡的 Elastic Beanstalk 环境。创建一个环境需要大约 5 分钟。Elastic Beanstalk 在创建运行应用程序所需的资源时，将输出信息性消息，并由 EB CLI 转发至终端。

4. 当环境创建过程完成时，通过运行 eb status 找到您的新环境的域名。

```
~/ebdjango$ eb status
Environment details for: django-env
  Application name: django-tutorial
  ...
  CNAME: eb-django-app-dev.elasticbeanstalk.com
  ...
```

您的环境的域名是 CNAME 属性的值。

5. 打开 `settings.py` 目录中的 `ebdjango` 文件。找到 `ALLOWED_HOSTS` 设置，然后将您在上一步中找到的应用程序域名添加到该设置的值。如果您在该文件中找不到此设置，请将此设置添加到一个新行。

```
...
ALLOWED_HOSTS = ['eb-django-app-dev.elasticbeanstalk.com']
```

6. 部署该文件，然后通过运行 eb deploy 部署您的应用程序。运行 eb deploy 时，EB CLI 会捆绑项目目录中的内容并将其部署到环境中。

```
~/ebdjango$ eb deploy
```

> **Note**
>
> 如果您将 Git 用于您的项目，请参阅将 EB CLI 与 Git 配合使用。

7. 当环境更新过程完成时，请使用 eb open 打开网站。

```
~/ebdjango$ eb open
```

这将使用为应用程序创建的域名打开一个浏览器窗口。您应该看到在本地创建和测试的相同 Django 网站。

如果您没有看到应用程序运行，或者出现错误消息，请查看排查部署问题以获取有关如何确定错误原因的帮助。

如果您确实 看到应用程序在运行，那么恭喜，您已使用 Elastic Beanstalk 部署了第一个 Django 应用程序！

更新应用程序

现在，您已在 Elastic Beanstalk 上运行应用程序，接下来可以更新和重新部署您的应用程序或其配置，Elastic Beanstalk 会处理更新实例和启动新的应用程序版本的工作。

在本示例中，我们将启用 Django 管理控制台并配置一些其他设置。

修改您的站点设置

默认情况下， Django 网站使用 UTC 时区显示时间。您可以通过在 `settings.py` 中指定时区来加以更改。

更改您的站点的时区

1. 在 `TIME_ZONE` 中修改 `settings.py` 设置。

   Example ~/ebdjango/ebdjango/settings.py

```
...
# Internationalization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'US/Pacific'
USE_I18N = True
```

```
USE_L10N = True
USE_TZ = True
```

如需获取时区列表，请访问此页面。

2. 将应用程序部署到您的 Elastic Beanstalk 环境。

```
~/ebdjango/$ eb deploy
```

创建站点管理员

您可以为 Django 应用程序创建站点管理员，以便从站点直接访问管理员控制台。管理员登录详细信息安全地存储在包含在 Django 生成的默认项目中包含的本地数据库镜像中。

创建站点管理员

1. 初始化 Django 应用程序的本地数据库。

```
(eb-virt) ~/ebdjango$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
```

2. 运行 `manage.py createsuperuser` 以创建管理员。

```
(eb-virt) ~/ebdjango$ python manage.py createsuperuser
Username: admin
```

```
Email address: me@mydomain.com
Password: ********
Password (again): ********
Superuser created successfully.
```

3. 要告诉 Django 在哪里存储静态文件，请在 STATIC_ROOT 中定义 settings.py。

   Example ~/ebdjango/ebdjango/settings.py

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_URL = '/static/'
STATIC_ROOT = 'static'
```

4. 运行 manage.py collectstatic 用管理员站点的静态资产（javascript、CSS 和图像）填充 static 目录。

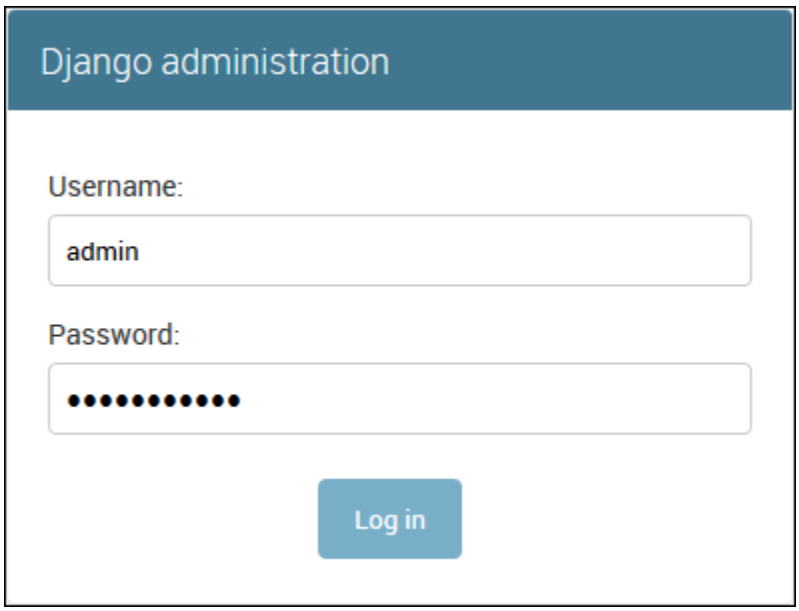
```
(eb-virt) ~/ebdjango$ python manage.py collectstatic
119 static files copied to ~/ebdjango/static
```

5. 部署您的应用程序。

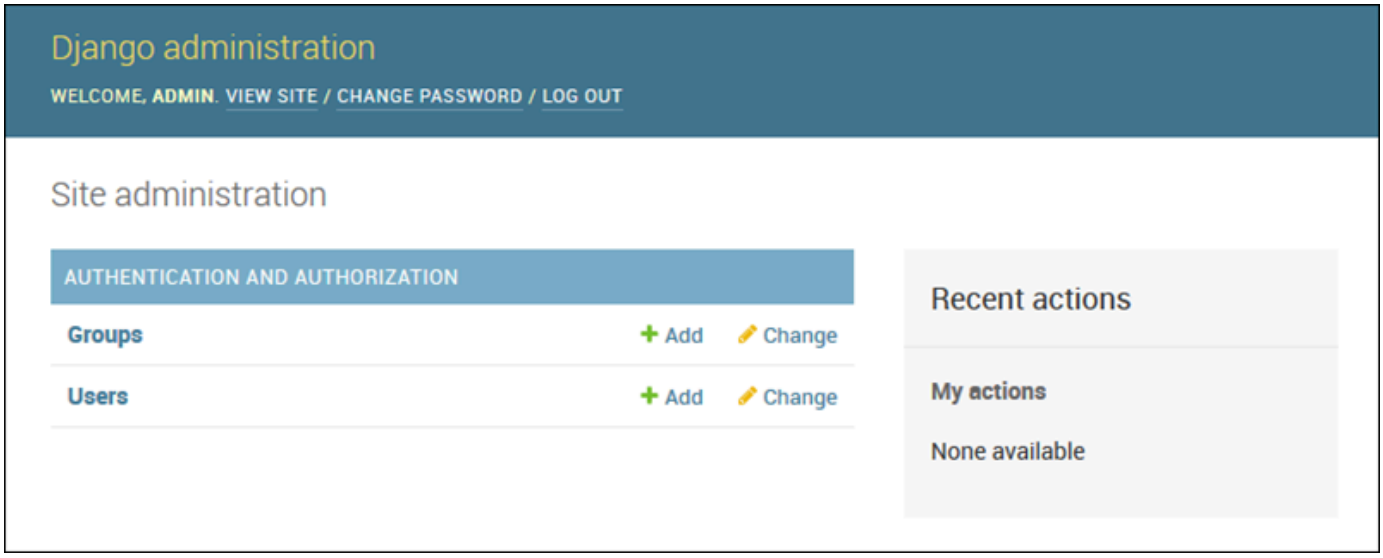
```
~/ebdjango$ eb deploy
```

6. 在您的浏览器中打开站点，并且将 /admin/ 附加到站点 URL，以此查看管理控制台，如下所示。

```
http://djang-env.p33kq46sfh.us-west-2.elasticbeanstalk.com/admin/
```



7. 使用您在步骤 2 中配置的用户名和密码登录。



您可以使用与本地更新/测试类似的过程，然后运行 eb deploy。Elastic Beanstalk 将执行更新您的实时服务器的工作，因此，您可以重点关注应用程序开发而不是服务器管理！

添加数据库迁移配置文件

您可以将命令添加到 `.ebextensions` 脚本，该脚本会在您的站点发生更新时运行。这样，您可以自动实现数据库迁移。

在部署应用程序时添加迁移步骤

1. 利用以下内容创建名为 db-migrate.config 的配置文件。

   Example ~/ebdjango/.ebextensions/db-migrate.config

   ```
   container_commands:
     01_migrate:
       command: "source /var/app/venv/*/bin/activate && python3 manage.py migrate"
       leader_only: true
   option_settings:
     aws:elasticbeanstalk:application:environment:
       DJANGO_SETTINGS_MODULE: ebdjango.settings
   ```

   此配置文件激活服务器的虚拟环境，在启动应用程序之前在部署流程中运行 manage.py migrate 命令。由于它在应用程序启动之前运行，因此您还必须明确配置 DJANGO_SETTINGS_MODULE 环境变量（通常在启动时由 wsgi.py 负责此项工作）。在命令中指定 leader_only: true 可确保当您重新部署到多个实例时，该命令仅运行一次。

2. 部署您的应用程序。

   ```
   ~/ebdjango$ eb deploy
   ```

清除

要在开发会话之间保存实例小时和其他 AWS 资源，请使用 eb terminate 终止 Elastic Beanstalk 环境。

```
~/ebdjango$ eb terminate django-env
```

此命令终止环境及其内部运行的所有 AWS 资源。但它不删除应用程序，因此，您始终可以再次运行 eb create，创建具有相同配置的更多环境。有关 EB CLI 命令的更多信息，请参阅使用 EB CLI 管理 Elastic Beanstalk 环境。

如果您不再使用示例应用程序，也可移除项目文件夹和虚拟环境。

```
~$ rm -rf ~/eb-virt
~$ rm -rf ~/ebdjango
```

后续步骤

有关 Django 的更多信息（包括详细教程），请访问官方文档。

如果您希望尝试其他 Python Web 框架，请参阅将 Flask 应用程序部署到 Elastic Beanstalk。

向 Python 应用程序环境中添加 Amazon RDS 数据库实例

您可以使用 Amazon Relational Database Service（Amazon RDS）数据库实例来存储由应用程序收集和修改的数据。数据库可以耦合到您的环境并由 Elastic Beanstalk 进行管理，也可以被创建为解耦数据库并由另一项服务进行外部管理。本主题提供使用 Elastic Beanstalk 控制台创建 Amazon RDS 的说明。数据库将耦合到您的环境并由 Elastic Beanstalk 进行管理。有关将 Amazon RDS 与 Elastic Beanstalk 集成的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

小节目录

向环境中添加数据库实例

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

添加一个数据库实例大约需要 10 分钟。环境更新完成后，您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息：

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

有关与 Elastic Beanstalk 环境耦合的数据库实例配置的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

下载驱动程序

向项目的需求文件中添加数据库驱动程序。

Example requirements.txt - 使用 MySQL 的 Django

```
Django==2.2
mysqlclient==2.0.3
```

Python 的常见驱动程序包

- MySQL – `mysqlclient`
- PostgreSQL – `psycopg2`
- Oracle – `cx_Oracle`
- SQL Server – `adodbapi`

有关更多信息，请参阅 Python DatabaseInterfaces 和 Django 2.2 - 支持的数据库。

连接到数据库

Elastic Beanstalk 在环境属性中提供所连数据库实例的连接信息。使用 `os.environ['VARIABLE']` 可读取这些属性并配置数据库连接。

Example Django 设置文件 - 数据库词典

```
import os

if 'RDS_HOSTNAME' in os.environ:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': os.environ['RDS_DB_NAME'],
            'USER': os.environ['RDS_USERNAME'],
            'PASSWORD': os.environ['RDS_PASSWORD'],
            'HOST': os.environ['RDS_HOSTNAME'],
            'PORT': os.environ['RDS_PORT'],
        }
    }
```

Python 工具和资源

在开发 Python 应用程序时，您可以在多个地方获取额外的帮助：

资源	描述
Boto（AWS SDK for Python）	使用 GitHub 安装 Boto。
Python 开发论坛	发布您的问题并获取反馈。

资源	描述
Python 开发者中心	示例代码、文档、工具和其他资源的一站式商店。

在 Elastic Beanstalk 上创建和部署 Ruby 应用程序

适用于 Ruby 的 AWS Elastic Beanstalk 可让您轻松地部署、管理和扩展使用亚马逊云科技的 Ruby Web 应用程序。Elastic Beanstalk 可供任何使用 Ruby 开发或托管 Web 应用程序的人员使用。本节逐步说明如何使用 Elastic Beanstalk 命令行界面（EB CLI）向 Elastic Beanstalk 部署示例应用程序，然后对该应用程序进行更新，以便使用 Rails 和 Sinatra Web 应用程序框架。

本章中的主题假设您对 Elastic Beanstalk 环境有所了解。如果您以前未使用过 Elastic Beanstalk，请尝试使用入门教程以了解基本知识。

主题

- 设置 Ruby 开发环境
- 使用 Elastic Beanstalk Ruby 平台
- 将 rails 应用程序部署到 Elastic Beanstalk
- 将 Sinatra 应用程序部署到 Elastic Beanstalk
- 向 Ruby 应用程序环境中添加 Amazon RDS 数据库实例

设置 Ruby 开发环境

设置 Ruby 开发环境以在本地测试应用程序，然后再将其部署到 AWS Elastic Beanstalk。本主题介绍开发环境设置步骤，并提供一些有用工具的安装页面链接。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

有关适用于所有语言的常见设置步骤和工具，请参阅配置用于 Elastic Beanstalk 的开发计算机

Sections

安装 Ruby

安装 GCC（如果没有 C 编译器）。在 Ubuntu 上，使用 `apt`。

```
~$ sudo apt install gcc
```

在 Amazon Linux 上，使用 `yum`。

```
~$ sudo yum install gcc
```

安装 RVM 以管理计算机上的 Ruby 语言安装。使用 rvm.io 处的命令获取项目密钥并运行安装脚本。

```
~$ gpg2 --recv-keys key1 key2
~$ curl -sSL https://get.rvm.io | bash -s stable
```

此脚本将 RVM 安装到用户目录中名为 `.rvm` 的文件夹中，并修改 shell 配置文件以在每次打开新终端时加载安装脚本。手动加载脚本以开始。

```
~$ source ~/.rvm/scripts/rvm
```

使用 `rvm get head` 获取最新版本。

```
~$ rvm get head
```

查看可用的 Ruby 版本。

```
~$ rvm list known
# MRI Rubies
...
[ruby-]2.6[.8]
[ruby-]2.7[.4]
```

```
[ruby-]3[.0.2]
...
```

检查《AWS Elastic Beanstalk 平台》文档中的 Ruby 以查找 Elastic Beanstalk 平台上可用的最新 Ruby 版本。安装该版本。

```
~$ rvm install 3.0.2
Searching for binary rubies, this might take some time.
Found remote file https://rubies.travis-ci.org/ubuntu/20.04/x86_64/ruby-3.0.2.tar.bz2
Checking requirements for ubuntu.
Updating system..
...
Requirements installation successful.
ruby-3.0.2 - #configure
ruby-3.0.2 - #download
...
```

测试 Ruby 安装。

```
~$ ruby --version
ruby 3.0.2p107 (2021-07-07 revision 0db68f0233) [x86_64-linux]
```

安装适用于 Ruby 的 S AWS DK

如果您需要从应用程序内部管理 AWS 资源，请安装 AWS SDK for Ruby。例如，借助 SDK for Ruby，您可以使用 Amazon DynamoDB（DynamoDB）来存储用户和会话信息，而无需创建关系数据库。

使用 gem 命令安装 SDK for Ruby 及其依赖项。

```
$ gem install aws-sdk
```

请访问 AWS SDK for Ruby 主页以了解更多信息和安装说明。

安装 IDE 或文本编辑器

集成开发环境（IDE）提供了便于应用程序开发的大量功能。如果你还没有使用 IDE 进行 Ruby 开发，可以试试 Aptana RubyMine ，看看哪种最适合你。

- 安装 Aptana

- RubyMine

> **Note**
>
> IDE 可以将您可能不希望提交到源代码控制的文件添加到项目文件夹中。要防止将这些文件提交到源代码控制，请使用 `.gitignore` 或您的源代码控制工具的同类功能。

如果您只是希望开始编码而不需要所有 IDE 功能，请考虑安装 Sublime Text。

使用 Elastic Beanstalk Ruby 平台

AWS Elastic Beanstalk Ruby 平台是一组环境配置，适用于可在 Puma 应用程序服务器下的 NGNIX 代理服务器后方运行的 Ruby Web 应用程序。每个平台分支对应于一个 Ruby 版本。如果使用了 RubyGems，您可以将 `Gemfile` 文件包含在源代码包中以在部署期间安装程序包。

应用程序服务器配置

Elastic Beanstalk 根据您在创建环境时选择的 Ruby 平台分支安装 Puma 应用程序服务器。有关 Ruby 平台版本提供的组件的更多信息，请参阅 AWS Elastic Beanstalk 平台指南中的支持的平台。

您可以使用自己提供的 Puma 服务器配置应用程序。这样就可选择使用除 Ruby 平台分支预装版本之外的 Puma 版本。您也可以将应用程序配置为使用其他应用程序服务器，例如 Passenger。为此，您必须在部署中包括并自定义 `Gemfile`。您还需要配置 `Procfile` 以启动应用程序服务器。有关更多信息，请参阅使用 Procfile 配置应用程序进程。

其他配置选项

Elastic Beanstalk 提供了配置选项，可供您用于自定义在 Elastic Beanstalk 环境中的 Amazon Elastic Compute Cloud（Amazon EC2）实例上运行的软件。您可配置应用程序所需的环境变量，启用日志轮换至 Amazon S3，并将应用程序源中包含静态文件的文件夹映射至代理服务器所提供的路径。该平台还预定义了一些与 Rails 和 Rack 相关的常见环境变量，便于发现和使用。

Elastic Beanstalk 控制台中提供了配置选项，可用于修改运行环境的配置。要避免在终止环境时丢失环境配置，可以使用保存的配置来保存您的设置，并在以后将这些设置应用到其他环境。

要保存源代码中的设置，您可以包含配置文件。在您每次创建环境或部署应用程序时，会应用配置文件中的设置。您还可在部署期间使用配置文件来安装程序包、运行脚本以及执行其他实例自定义操作。

在 Elastic Beanstalk 控制台中应用的设置会覆盖配置文件中的相同设置（如果存在）。这让您可以在配置文件中包含默认设置，并使用控制台中的特定环境设置加以覆盖。有关优先顺序和其他设置更改方法的更多信息，请参阅配置选项。

有关扩展 Elastic Beanstalk 基于 Linux 的平台的各种方法的详细信息，请参阅the section called “扩展 Linux 平台”。

配置 Ruby 环境

您可以使用 Elastic Beanstalk 控制台启用到 Amazon S3 的日志轮换，并配置应用程序可从环境中读取的变量。

访问相应环境的软件配置设置

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

日志选项

Log options（日志选项）部分有两个设置：

- Instance profile（实例配置文件）– 指定有权访问与应用程序关联的 Amazon S3 存储桶的实例配置文件。
- Enable log file rotation to Amazon S3（启用到 Amazon S3 的日志轮换）– 指定是否将应用程序的 Amazon EC2 实例的日志文件复制到与应用程序关联的 Amazon S3 存储桶。

静态文件

为了提高性能，您可以使用 Static files（静态文件）部分配置代理服务器，以便从 Web 应用程序内的一组目录提供静态文件（例如 HTML 或图像）。对于每个目录，您都将虚拟路径设置为目录映射。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

有关使用配置文件或 Elastic Beanstalk 控制台配置静态文件的详细信息，请参阅 the section called “静态文件”。

默认情况下，Ruby 环境中的代理服务器配置为提供静态文件，如下所示：

- public 文件夹中的文件通过 /public 路径和域根目录（/ 路径）提供。
- public/assets 子文件夹中的文件通过 /assets 路径提供。

以下示例说明默认配置的工作原理：

- 如果您的应用程序源代码在名为 public 的文件夹中包含一个名为 logo.png 的文件，则代理服务器将通过 *subdomain*.elasticbeanstalk.com/public/logo.png 和 *subdomain*.elasticbeanstalk.com/logo.png 将其提供给用户。
- 如果您的应用程序源代码在 public 文件夹内名为 assets 的文件夹中包含一个名为 logo.png 的文件，则代理服务器将通过 *subdomain*.elasticbeanstalk.com/assets/logo.png 将其提供给用户。

您可以配置静态文件的其他映射。有关更多信息，请参阅本主题后面的 Ruby 配置命名空间。

> **Note**
>
> 对于 Ruby 2.7 AL2 版本 3.3.7 之前的平台版本，默认 Elastic Beanstalk nginx 代理服务器配置不支持通过域根目录 (*subdomain*.elasticbeanstalk.com/) 提供静态文件。此平台版本于 2021 年 10 月 21 日发布。有关更多信息，请参阅 AWS Elastic Beanstalk 发布说明中的新平台版本 - Ruby。

环境属性

在环境属性部分，您可以在运行应用程序的 Amazon EC2 实例上指定环境配置设置。环境属性会以密钥值对的形式传递到应用程序。

Ruby 平台为环境配置定义了以下属性：

- BUNDLE_WITHOUT – 从 Gemfile 安装依赖项时所忽略的组的冒号分隔列表。
- BUNDLER_DEPLOYMENT_MODE – 设置为 true（默认值）以使用捆绑程序在部署模式下安装依赖项。设置为 false 以在开发模式中运行 bundle install。

> **Note**
>
> 此环境属性未在 Amazon Linux AMI Ruby 平台分支（在 Amazon Linux 2 之前）上定义。

- RAILS_SKIP_ASSET_COMPILATION – 设置为 `true` 可在部署期间跳过运行 `rake assets:precompile`。
- RAILS_SKIP_MIGRATIONS – 设置为 `true` 可在部署期间跳过运行 `rake db:migrate`。
- RACK_ENV – 指定 Rack 的环境阶段。例如，`development`、`production` 或 `test`。

在运行于 Elastic Beanstalk 中的 Ruby 环境内，可通过使用 ENV 对象访问环境变量。例如，您可以使用以下代码将名为 API_ENDPOINT 的属性读取到某个变量：

```
endpoint = ENV['API_ENDPOINT']
```

参阅 环境属性和其他软件设置 了解更多信息。

Ruby 配置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

您可以使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间来配置环境代理以提供静态文件。您可以定义虚拟路径到应用程序目录的映射。

Ruby 平台不定义任何特定于平台的命名空间。相反，它为常见的 Rails 和 Rack 选项定义环境属性。

以下配置文件指定一个静态文件选项，该选项将名为 `staticimages` 的目录映射到路径 `/images`，设置每个平台定义的环境属性，还设置名为 LOGGING 的附加环境属性。

Example .ebextensions/ruby-settings.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /images: staticimages
  aws:elasticbeanstalk:application:environment:
    BUNDLE_WITHOUT: test
    BUNDLER_DEPLOYMENT_MODE: true
    RACK_ENV: development
    RAILS_SKIP_ASSET_COMPILATION: true
```

```
    RAILS_SKIP_MIGRATIONS: true
    LOGGING: debug
```

> **Note**
>
> BUNDLER_DEPLOYMENT_MODE 环境属性和 aws:elasticbeanstalk:environment:proxy:staticfiles 命名空间未在 Amazon Linux AMI Ruby 平台分支（在 Amazon Linux 2 之前）上定义。

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

使用 Gemfile 安装程序包

借助位于项目源的根目录中的 Gemfile 文件，您可使用 RubyGems 来安装应用程序所需的程序包。

Example Gemfile

```
source "https://rubygems.org"
gem 'sinatra'
gem 'json'
gem 'rack-parser'
```

如果 Gemfile 文件存在，Elastic Beanstalk 将运行 bundle install 以安装依赖项。有关更多信息，请参阅 Bundler.io 网站上的 Gemfiles 和 Bundle（捆绑包）页面。

> **Note**
>
> 除了 Ruby 平台预装的默认版本外，还可以使用其他版本的 Puma。为此，请在 Gemfile 中包含指定版本的条目。您还可以使用自定义的 Gemfile 来指定不同的应用程序服务器，例如 Passenger。
> 对于这两种情况，您都需要配置 Procfile 以启动应用程序服务器。
> 有关更多信息，请参阅使用 Procfile 配置应用程序进程。

使用 Procfile 配置应用程序进程

要指定启动 Ruby 应用程序的命令，请在源代码包的根目录中包含一个名为 Procfile 的文件。

> **Note**
>
> Elastic Beanstalk 在 Amazon Linux AMI Ruby 平台分支（在 Amazon Linux 2 之前）上不支持此功能。无论 Ruby 版本为何，名称包含 with Puma 或 with Passenger 的平台分支都早于 Amazon Linux 2，并且不支持 `Procfile` 属性。

有关编写和使用 `Procfile` 的详细信息，请展开the section called “扩展 Linux 平台”中的 Buildfile 和 Procfile 部分 。

如果您不提供 `Procfile`，则 Elastic Beanstalk 会生成以下默认文件，该文件假定您使用的是预安装的 Puma 应用程序服务器。

```
web: puma -C /opt/elasticbeanstalk/config/private/pumaconf.rb
```

如果您想使用自己提供的 Puma 服务器，则可以使用 Gemfile 进行安装。以下示例 `Procfile` 演示了如何启动它。

Example Procfile

```
web: bundle exec puma -C /opt/elasticbeanstalk/config/private/pumaconf.rb
```

如果要使用 Passenger 应用程序服务器，请使用以下示例文件配置 Ruby 环境，以安装和使用 Passenger。

1. 使用此示例文件安装 Passenger。

   Example Gemfile

   ```
   source 'https://rubygems.org'
   gem 'passenger'
   ```

2. 使用此示例文件指示 Elastic Beanstalk 启动 Passenger。

   Example Procfile

   ```
   web: bundle exec passenger start /var/app/current --socket /var/run/puma/my_app.sock
   ```

> **Note**
>
> 您无需在 nginx 代理服务器的配置中更改任何内容即可使用 Passenger。要使用其他应用程序服务器，您可能需要自定义 nginx 配置以将请求正确转发到您的应用程序。

将 rails 应用程序部署到 Elastic Beanstalk

Rails 是一个适用于 Ruby 的开源 model-view-controller (MVC) 框架。本教程将引导您完成生成 Rails 应用程序并将其部署到 AWS Elastic Beanstalk 环境中的过程。

Sections

先决条件

Elastic Beanstalk 基础知识

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

命令行

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

Rails 依赖项

Rails 框架 6.1.4.1 具有以下依赖项。请确保您已安装所有依赖项。

- Ruby 2.5.0 或更高版本 - 有关安装说明，请参阅 设置 Ruby 开发环境。

  在本教程中，我们使用 Ruby 3.0.2 和相应的 Elastic Beanstalk 平台版本。
- Node.js – 有关安装说明，请参阅通过软件包管理器安装 Node.js。
- Yarn – 有关安装说明，请参阅 Yarn 网站上的安装。

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 Ruby 平台并接受默认设置和示例代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。
4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。

- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

安装 rails 并生成网站

使用 `gem` 命令安装 Rails 及其依赖项。

```
~$ gem install rails
Fetching: concurrent-ruby-1.1.9.gem
Successfully installed concurrent-ruby-1.1.9
Fetching: rack-2.2.3.gem
```

```
Successfully installed rack-2.2.3
...
```

测试 Rails 安装。

```
~$ rails --version
Rails 6.1.4.1
```

将 `rails new` 与应用程序的名称一起使用以创建新 Rails 项目。

```
~$ rails new ~/eb-rails
```

Rails 将创建具有指定名称的目录，生成在本地运行示例项目所需的所有文件，然后运行捆绑程序以安装在项目的 Gemfile 中定义的所有依赖项（Gems）。

> **Note**
>
> 此过程将为该项目安装最新的 Puma 版本。此版本可能与 Elastic Beanstalk 在环境的 Ruby 平台版本上提供的版本不同。要查看 Elastic Beanstalk 提供的 Puma 版本，请参阅《AWS Elastic Beanstalk 平台指南》中的 Ruby 平台历史记录。有关最新 Puma 版本的更多信息，请参阅 Puma.io 网站。如果这两个 Puma 版本不一致，请使用下面的任意一种选项：
>
> - 使用之前 `rails new` 命令安装的 Puma 版本。在这种情况下，您必须为平台添加一个 `Procfile` 以使用自己提供的 Puma 服务器版本。有关更多信息，请参阅 使用 Procfile 配置应用程序进程。
> - 更新 Puma 版本，使其与环境的 Ruby 平台版本中预安装的版本保持一致。为此，请在 Gemfile 中修改 Puma 版本，该文件位于项目源目录的根中。然后运行 `bundle update`。有关更多信息，请参阅 Bundler.io 网站上的 bundle update（捆绑包更新）页面。

通过在本地运行默认项目测试 Rails 安装。

```
~$ cd eb-rails
~/eb-rails$ rails server
=> Booting Puma
=> Rails 6.1.4.1 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 5.5.2 (ruby 3.0.2-p107) ("Zawgyi")
```

```
*  Min threads: 5
*  Max threads: 5
*  Environment: development
*          PID: 77857
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
...
```

在 Web 浏览器中打开 `http://localhost:3000` 以查看运转中的默认项目。

此页面仅在开发模式中可见。向应用程序的首页添加一些内容来支持至 Elastic Beanstalk 的生产部署。使用 `rails generate` 为欢迎页面创建控制器、路由和视图。

```
~/eb-rails$ rails generate controller WelcomePage welcome
      create  app/controllers/welcome_page_controller.rb
       route  get 'welcome_page/welcome'
      invoke  erb
      create    app/views/welcome_page
```

```
      create    app/views/welcome_page/welcome.html.erb
      invoke  test_unit
      create    test/controllers/welcome_page_controller_test.rb
      invoke  helper
      create    app/helpers/welcome_page_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/welcome_page.coffee
      invoke    scss
      create      app/assets/stylesheets/welcome_page.scss.
```

这将提供访问 /welcome_page/welcome 处的页面所需的一切。但是，在发布更改之前，请更改视图中的内容并添加路由以使此页面显示在网站的顶层。

使用文本编辑器编辑 app/views/welcome_page/welcome.html.erb 中的内容。在本示例中，您将使用 cat 覆盖现有文件的内容。

Example app/views/welcome_page/welcome.html.erb

```
<h1>Welcome!</h1>
<p>This is the front page of my first Rails application on Elastic Beanstalk.</p>
```

最后，将以下路由添加到 config/routes.rb：

Example config/routes.rb

```
Rails.application.routes.draw do
  get 'welcome_page/welcome'
  root 'welcome_page#welcome'
```

这将指示 Rails 将对网站的根目录的请求路由到欢迎页面控制器的欢迎方法，该方法将在欢迎视图中呈现内容（welcome.html.erb）。

要使 Elastic Beanstalk 在 Ruby 平台上成功部署应用程序，我们需要更新 Gemfile.lock。Gemfile.lock 的一些依赖项可能是特定于平台的。因此，我们需要将 **platform ruby** 添加到 Gemfile.lock，以使所有必需的依赖项均随部署一起安装。

Example

```
~/eb-rails$ bundle lock --add-platform ruby
```

```
Fetching gem metadata from https://rubygems.org/.............
Resolving dependencies...
Writing lockfile to /Users/janedoe/EBDPT/RubyApps/eb-rails-doc-app/Gemfile.lock
```

配置 rails 设置

使用 Elastic Beanstalk 控制台为 Rails 配置环境属性。将 SECRET_KEY_BASE 环境属性设置为最多 256 个字母数字字符的字符串。

Rails 使用此属性来创建密钥。因此，您应该保密，不要将密钥以纯文本形式存储在源代码控制中。相反，您应通过环境属性在环境中向 Rails 代码提供该密钥。

在 Elastic Beanstalk 控制台中配置环境属性

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 向下滚动到环境属性。
6. 选择添加环境属性。
7. 输入属性名称和值对。
8. 如需添加更多变量，请重复步骤 6 和步骤 7。
9. 要保存更改，请选择页面底部的 Apply（应用）。

现在准备好将站点部署到环境中。

部署您的应用程序

创建包含 Rails 所创建文件的源包。以下命令将创建名为 `rails-default.zip` 的源包。

```
~/eb-rails$ zip ../rails-default.zip -r * .[^.]*
```

将源包上传到 Elastic Beanstalk 以将 Rails 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

后续步骤

有关 Rails 的更多信息，请访问 rubyonrails.org。

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

将 Sinatra 应用程序部署到 Elastic Beanstalk

本演练介绍如何将简单的 Sinatra Web 应用程序部署到 AWS Elastic Beanstalk。

先决条件

本教程假设您对基本 Elastic Beanstalk 操作和 Elastic Beanstalk 控制台有一定了解。如果尚不了解，请按照开始使用 Elastic Beanstalk中的说明操作以启动您的第一个 Elastic Beanstalk 环境。

为了遵循本指南中的步骤，您需要命令行终端或 Shell，以便运行命令。命令显示在列表中，以提示符（$）和当前目录名称（如果有）开头。

```
~/eb-project$ this is a command
this is output
```

在 Linux 和 macOS 中，您可使用您首选的 Shell 和程序包管理器。在 Windows 上，你可以安装适用于 Linux 的 Windows 子系统来获得与 Windows 集成的版本的 Ubuntu 和 Bash。

Sinatra 2.1.0 需要 Ruby 2.3.0 或更高版本。在本教程中，我们使用 Ruby 3.0.2 和相应的 Elastic Beanstalk 平台版本。按照设置 Ruby 开发环境上的说明操作来安装 Ruby。

启动 Elastic Beanstalk 环境

使用 Elastic Beanstalk 控制台创建 Elastic Beanstalk 环境。选择 Ruby 平台并接受默认设置和示例代码。

启动环境（控制台）

1. 使用以下预配置链接打开 Elastic Beanstalk 控制台：console.aws.amazon.com/elasticbeanstalk/home#/ newApplicationName=Tutorials &environtype= LoadBalanced
2. 对于 Platform（平台），选择与应用程序使用的语言匹配的平台和平台分支。
3. 对于应用程序代码，选择示例应用程序。

4. 选择复查并启动。
5. 检查可用选项。选择要使用的可用选项，然后在准备就绪后选择创建应用程序。

环境创建大约需要 5 分钟，将创建以下资源：

- EC2 实例 – 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

所有这些资源都由 Elastic Beanstalk 管理。当您终止环境时，Elastic Beanstalk 会终止其包含的所有资源。

> **Note**
>
> Elastic Beanstalk 创建的 Amazon S3 存储桶将在多个环境之间共享并且在环境终止期间不会被删除。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon S3 结合使用。

编写基本 sinatra 网站

创建和部署 sinatra 应用程序

1. 利用以下内容创建名为 config.ru 的配置文件。

   Example config.ru

   ```
   require './helloworld'
   run Sinatra::Application
   ```

2. 使用以下内容创建名为 helloworld.rb 的 Ruby 代码文件。

   Example helloworld.rb

   ```
   require 'sinatra'
   get '/' do
     "Hello World!"
   end
   ```

3. 使用以下内容创建 Gemfile。

   Example Gemfile

   ```
   source 'https://rubygems.org'
   gem 'sinatra'
   gem 'puma'
   ```

4. 运行捆绑安装以生成 `Gemfile.lock`

   Example

   ```
   ~/eb-sinatra$ bundle install
   ```

```
Fetching gem metadata from https://rubygems.org/....
Resolving dependencies...
Using bundler 2.2.22
Using rack 2.2.3
...
```

5. 要使 Elastic Beanstalk 在 Ruby 平台上成功部署应用程序，我们需要更新 `Gemfile.lock`。`Gemfile.lock` 的一些依赖项可能是特定于平台的。因此，我们需要将 **`platform ruby`** 添加到 `Gemfile.lock`，以使所有必需的依赖项均随部署一起安装。

   Example

   ```
   ~/eb-sinatra$ bundle lock --add-platform ruby
   Fetching gem metadata from https://rubygems.org/....
   Resolving dependencies...
   Writing lockfile to /Users/janedoe/EBDPT/RubyApps/eb-sinatra/Gemfile.lock
   ```

6. 使用以下内容创建 Procfile。

   Example Procfile

   ```
   web: bundle exec puma -C /opt/elasticbeanstalk/config/private/pumaconf.rb
   ```

部署您的应用程序

创建包含您的源文件的源包。以下命令将创建名为 `sinatra-default.zip` 的源包。

```
~/eb-sinatra$ zip ../sinatra-default.zip -r * .[^.]*
```

将源包上传到 Elastic Beanstalk 以将 Sinatra 部署到您的环境。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

清除

Elastic Beanstalk 使用完毕时，您可以终止您的环境。Elastic Beanstalk AWS 会终止与您的环境关联的所有资源，例如 Amazon EC2 实例、数据库实例、负载均衡器、安全组和警报。

从控制台终止你的 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

利用 Elastic Beanstalk，可以随时为您的应用程序轻松创建新环境。

后续步骤

有关 Sinatra 的更多信息，请访问 sinatrarb.com。

当您继续开发应用程序时，您可能希望通过某种方式来管理环境和部署应用程序，而无需手动创建 .zip 文件并将该文件上传到 Elastic Beanstalk 控制台。Elastic Beanstalk 命令行界面 (EB CLI easy-to-use) 提供了从命令行创建、配置应用程序并将其部署到 Elastic Beanstalk 环境的命令。

最后，如果计划在生产环境中使用应用程序，您会希望为环境配置自定义域名，并为安全连接启用 HTTPS。

向 Ruby 应用程序环境中添加 Amazon RDS 数据库实例

您可以使用 Amazon Relational Database Service（Amazon RDS）数据库实例来存储由应用程序收集和修改的数据。数据库可以耦合到您的环境并由 Elastic Beanstalk 进行管理，也可以被创建为解耦

数据库并由另一项服务进行外部管理。本主题提供使用 Elastic Beanstalk 控制台创建 Amazon RDS 的说明。数据库将耦合到您的环境并由 Elastic Beanstalk 进行管理。有关将 Amazon RDS 与 Elastic Beanstalk 集成的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

小节目录

向环境中添加数据库实例

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

添加一个数据库实例大约需要 10 分钟。环境更新完成后，您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息：

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。

属性名称	描述	属性值
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

有关与 Elastic Beanstalk 环境耦合的数据库实例配置的更多信息，请参阅 将数据库添加到 Elastic Beanstalk 环境。

下载适配器

向项目的 gem 文件中添加数据库适配器。

Example Gemfile – 使用 MySQL 的 Rails

```
source 'https://rubygems.org'
gem 'puma'
gem 'rails', '~> 6.1.4', '>= 6.1.4.1'
gem 'mysql2'
```

Ruby 的常见适配器 Gem

- MySQL – mysql2
- PostgreSQL – pg

- Oracle – activerecord-oracle_enhanced-adapter
- SQL Server – activerecord-sqlserver-adapter

连接到数据库

Elastic Beanstalk 在环境属性中提供所连数据库实例的连接信息。使用 ENV['*VARIABLE*'] 可读取这些属性并配置数据库连接。

Example config/database.yml - Ruby on rails 数据库配置（MySQL）

```
production:
  adapter: mysql2
  encoding: utf8
  database: <%= ENV['RDS_DB_NAME'] %>
  username: <%= ENV['RDS_USERNAME'] %>
  password: <%= ENV['RDS_PASSWORD'] %>
  host: <%= ENV['RDS_HOSTNAME'] %>
  port: <%= ENV['RDS_PORT'] %>
```

教程和示例

特定于语言和框架的教程分布在《 AWS Elastic Beanstalk 开发者指南》中。新的和更新的教程在发布后将添加到此列表。首先显示最近更新。

这些教程面向中间用户，并且可能不包含基本步骤（例如注册 AWS）的说明。如果这是你第一次使用 AWS Elastic Beanstalk，请查看入门指南，让你的第一个 Elastic Beanstalk 环境启动并运行。

- Ruby on Rails – 将 rails 应用程序部署到 Elastic Beanstalk
- Ruby and Sinatra – 将 Sinatra 应用程序部署到 Elastic Beanstalk
- PHP 和 MySQL HA 配置 – 将带有外部 Amazon RDS 数据库的高可用性 PHP 应用程序部署到 Elastic Beanstalk
- PHP 和 Laravel – 将 Laravel 应用程序部署到 Elastic Beanstalk
- PHP 和 CakePHP – 将 CakePHP 应用程序部署到 Elastic Beanstalk
- PHP 和 Drupal HA 配置 – 将带有外部 Amazon RDS 数据库的高可用性 Drupal 网站部署到 Elastic Beanstalk
- PHP 和 WordPress HA 配置-将带有外部 Amazon RDS 数据库的高可用性 WordPress 网站部署到 Elastic Beanstalk
- Node.js 和 DynamoDB HA 配置 – 将带 DynamoDB 的 Node.js 应用程序部署到 Elastic Beanstalk
- ASP.NET 内核 – 教程：使用 Elastic Beanstalk 部署 ASP.NET Core 应用程序
- Python 和 Flask – 将 Flask 应用程序部署到 Elastic Beanstalk
- Python 和 Django – 将 Django 应用程序部署到 Elastic Beanstalk
- Node.js 和 Express – 将 Express 应用程序部署到 Elastic Beanstalk
- Docker、PHP 和 nginx – 使用 Elastic Beanstalk 控制台启动 ECS 托管式 Docker 环境

通过以下链接，您可以下载创建环境时 Elastic Beanstalk 使用的示例应用程序，而无需提供源包：

- Docker – docker.zip
- 多容器 Docker — 2.zip docker-multicontainer-v
- 预配置的 Docker (Glassfish) — 1.zip docker-glassfish-v
- Go – go.zip
- Corretto – corretto.zip
- Tomcat – tomcat.zip

- Linux 上的.NET 内核 — dotnet-core-linux.zi
- .NET 核心 — dotnet-asp-windows.zip
- Node.js – nodejs.zip
- PHP – php.zip
- Python – python.zip
- Ruby – ruby.zip

更多展示其他 Web 框架、库和工具使用情况的示例应用程序可作为开源项目获得，网址为 GitHub：

- 负载平衡 WordPress（教程）— 用于在负载平衡的 Elastic Beanstalk 环境中 WordPress 安全安装和运行的配置文件。
- Load Balanced Drupal（教程）– 用于安全地安装 Drupal 并在负载均衡的 Elastic Beanstalk 环境中运行 Drupal 的配置文件和说明。
- Scorek eep-RESTful Web API，它使用 Spring 框架和提供用于创建和管理用户、会话和游戏的界面。 AWS SDK for Java 此 API 与通过 HTTP 使用它的 Angular 1.5 Web 应用程序捆绑在一起。包括显示与 Amazon Cognito 集成的分支和亚马逊 Relational AWS X-RayDatabase Service。

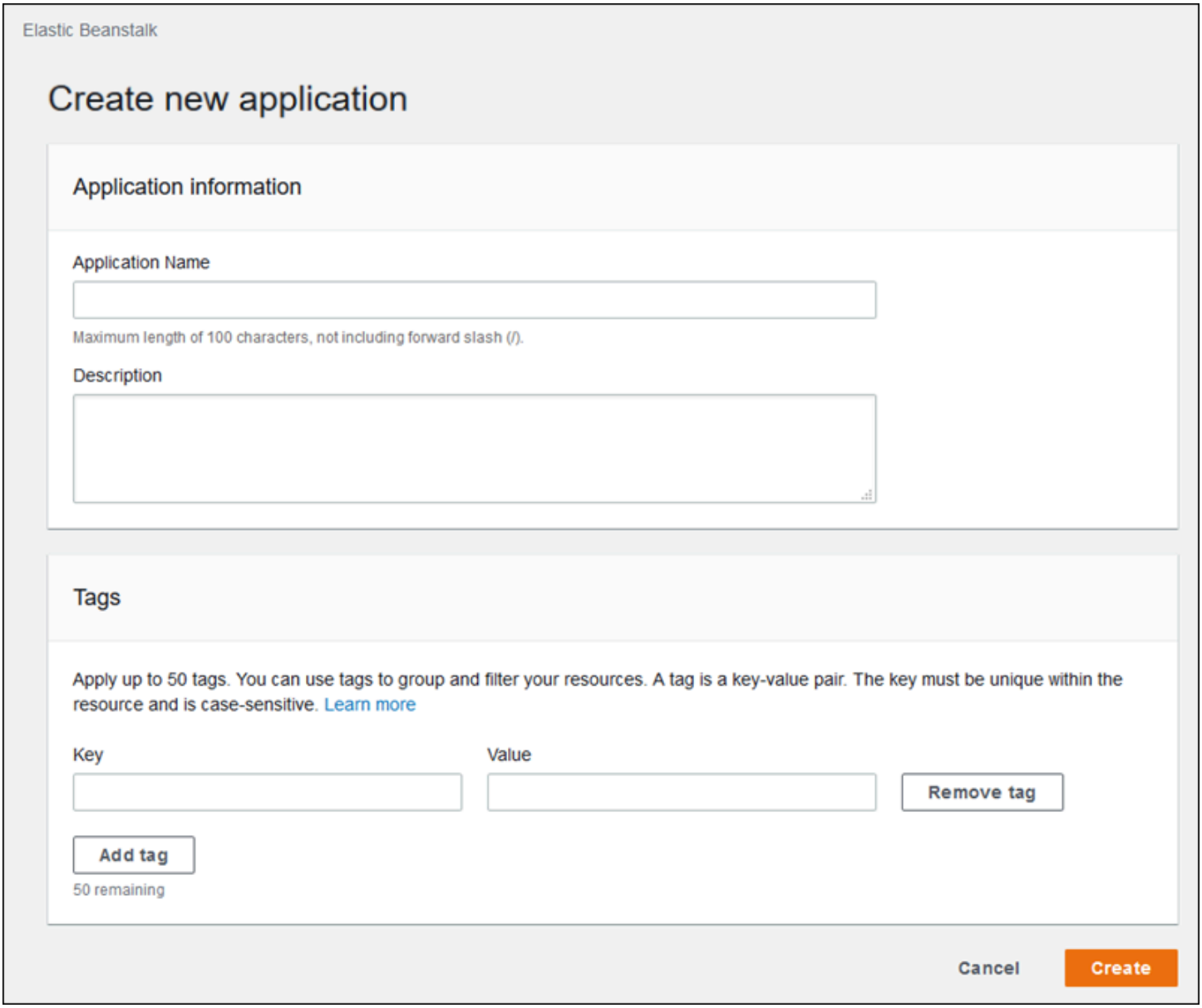
  应用程序使用 Java SE 平台的功能下载依赖项和用作构建基础的实例，从而最小化源包的大小。此外，应用程序还包含覆盖默认配置的 nginx 配置文件，以通过代理在端口 80 上静态地为前端 Web 应用程序提供服务，并将针对 `/api` 下路径的请求路由到在 `localhost:5000` 上运行的 API。
- 它有蛇吗？ -显示在 Elastic Beanstalk 的 Java EE Web 应用程序中使用 RDS 的 Tomcat 应用程序。项目显示 Servlets、JSPs、Simple Tag Support、Tag Files、JDBC、SQL、Log4J、Bootstrap、Jackson 和 Elastic Beanstalk 配置文件的使用。
- Locust Load Generator - 此项目显示安装和运行 Locust（一种以 Python 编写的负载生成工具）的 Java SE 平台功能使用。项目包含安装和配置 Locust (配置 DynamoDB 的构建脚本) 和运行 Locust 的 Procfile 的配置文件。
- 分享您的想法（教程）– 显示 MySQL 上的 Amazon RDS、Composer 和配置文件的使用的 PHP 应用程序。
- 新启动（教程）-Node.js 示例应用程序，它展示了 DynamoDB、Node.js 中的 SDK JavaScript 、AWS npm 包管理和配置文件的用法。

管理和配置 Elastic Beanstalk 应用程序

使用 AWS Elastic Beanstalk 的第一步是创建一个应用程序，该应用程序表示AWS中的 Web 应用程序。在 Elastic Beanstalk 中，应用程序充当一种容器，用于容纳运行您的 Web 应用程序的环境、Web 应用程序的源代码的版本、已保存的配置、日志以及使用 Elastic Beanstalk 时创建的其他构件。

创建应用程序

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后选择 Create application（创建应用程序）。
3. 使用屏幕上的表单提供应用程序名称。
4. （可选）提供说明，并添加标签键和值。
5. 选择 Create（创建）。



创建应用程序后，控制台会提示您为其创建一个环境。有关所有可用选项的详细信息，请参阅创建 Elastic Beanstalk 环境。

如果您不再需要某个应用程序，则可删除它。

> **Warning**
>
> 删除某个应用程序将终止所有关联的环境并删除属于该应用程序的所有应用程序版本和保存的配置。

删除应用程序

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications (应用程序)，然后在列表中选择您的应用程序。
3. 选择 Actions (操作)，然后选择 Delete application (删除应用程序)。

主题

Elastic Beanstalk 应用程序管理控制台

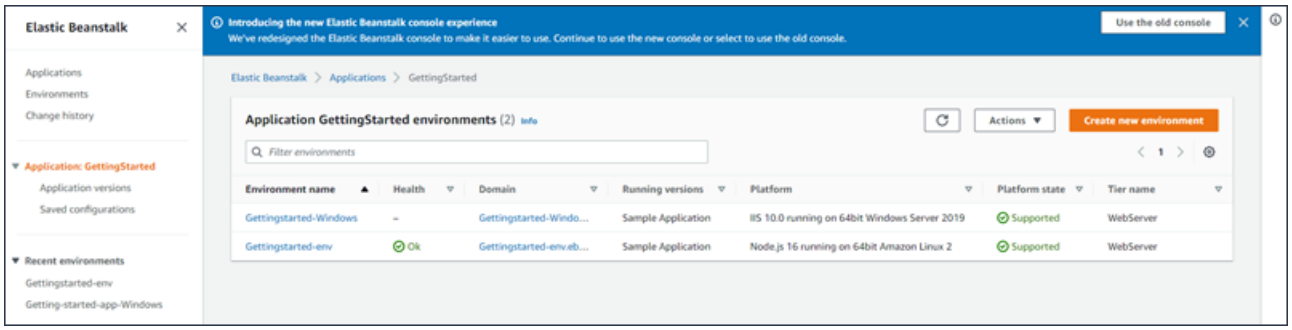
您可以使用 AWS Elastic Beanstalk 控制台管理应用程序、应用程序版本和保存的配置。

访问应用程序管理控制台

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

应用程序概述页面显示了一个列表，其中概述了与应用程序关联的所有环境。



3. 可以通过几种方法来继续操作：

a. 选择 Actions (操作) 下拉菜单，然后选择其中一个应用程序管理操作。要在此应用程序中启动环境，您可以直接选择 Create a new environment (创建新环境)。有关详细信息，请参阅 the section called “创建环境”。

b. 选择环境名称以转至该环境的环境管理控制台，可在其中配置、监控或管理该环境。

c. 在导航窗格中选择应用程序名称后面的 Application versions (应用程序版本)，以查看和管理应用程序的应用程序版本。

   应用程序版本是应用程序代码的上传版本。您可以上传新版本、将现有版本部署到任何应用程序环境或删除旧版本。有关更多信息，请参阅管理应用程序版本。

d. 在导航窗格中选择应用程序名称后面的 Saved configurations (已保存配置)，以查看和管理从正在运行的环境中保存的配置。

   保存的配置是一组设置，可用于将环境的设置还原到上一个状态或创建具有相同设置的环境。有关更多信息，请参阅使用 Elastic Beanstalk 保存的配置。

管理应用程序版本

只要您上传源代码，Elastic Beanstalk 就会创建应用程序版本。当您使用环境管理控制台或 EB CLI 创建环境或上传并部署代码时，通常会出现此情况。Elastic Beanstalk 会根据应用程序的生命周期策略以及当您删除应用程序时删除这些应用程序版本。有关应用程序生命周期策略的详细信息，请参阅配置应用程序版本生命周期设置。

您也可以上传源包，而无需从应用程序管理控制台或使用 EB CLI 命令 eb appversion 部署它。Elastic Beanstalk 将源包存储在 Amazon Simple Storage Service (Amazon S3) 中且不自动删除它们。

您可以在创建应用程序版本和编辑现有应用程序版本的标签时向其应用标签。有关详细信息，请参阅标记应用程序版本。

创建新应用程序版本

您还可以使用 EB CLI 创建新的应用程序版本。有关更多信息，请参阅 EB CLI 命令章节中的 eb appversion。

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Application versions（应用程序版本）。
4. 请选择 Upload（上传）。使用屏幕上的表单上传应用程序的源包。

> **Note**
> 源包的文件大小限制为 500 MB。

5. （可选）提供简要说明，并添加标签键和值。
6. 请选择 Upload（上传）。

您指定的文件会与您的应用程序关联。您可以将应用程序版本部署到新的或现有环境。

随着时间的推移，应用程序可能会积累许多应用程序版本。要节省存储空间并避免达到应用程序版本配额，最好是删除不再需要的应用程序版本。

> **Note**
> 删除应用程序版本不会影响当前正在运行该版本的环境。

删除应用程序版本

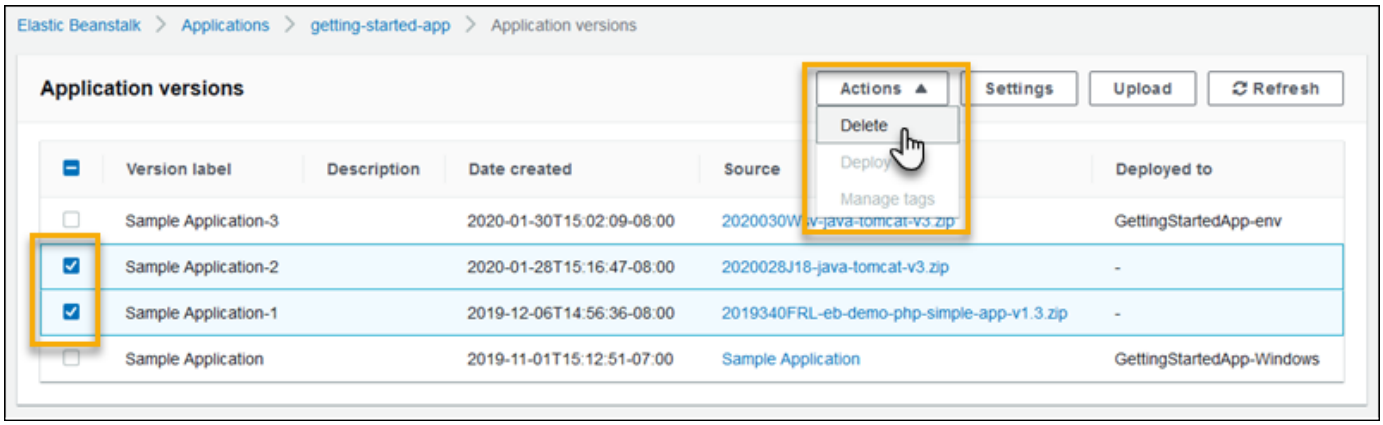
您还可以使用 EB CLI 删除应用程序版本。有关更多信息，请参阅 EB CLI 命令章节中的 eb appversion。

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Application versions（应用程序版本）。

4. 选择要删除的一个或多个应用程序版本。



5. 选择 Actions (操作)，然后选择 Delete (删除)。

6. （可选）Amazon Simple Storage Service (Amazon S3) 存储桶中保留这些应用程序版本的应用程序源包，请取消选中 Delete versions from Amazon S3 (从 Amazon S3 中删除版本) 框。

7. 选择 Delete (删除)。

您还可以通过配置应用程序版本生命周期设置来将 Elastic Beanstalk 配置为自动删除旧版本。如果您配置了这些应用程序生命周期设置，它们将在您创建新的应用程序版本时应用。例如，如果您配置最多 25 个应用程序版本，那么当您上传第 26 个版本时，Elastic Beanstalk 将删除最旧的版本。如果您设置的最长使用期为 90 天，在上传新版本时，任何超过 90 天的版本都将被删除。有关详细信息，请参阅 the section called “版本生命周期”。

如果您未选择从 Amazon S3 中删除源包，则 Elastic Beanstalk 仍会从其记录中删除该版本。但是，源包保留在您的 Elastic Beanstalk 存储桶中。应用程序版本配额仅适用于 Elastic Beanstalk 跟踪的版本。因此，您可以删除版本以保持在配额内，但将所有源包保留在 Amazon S3 中。

> **Note**
>
> 此应用程序版本配额不适用于源包，但您可能仍会产生 Amazon S3 费用，并且在需要个人信息的时间范围之外保留这类信息。Elastic Beanstalk 一定不会自动删除源包。当您不再需要源包时，应将其删除。

配置应用程序版本生命周期设置

每次您使用 Elastic Beanstalk 控制台或 EB CLI 上传应用程序的新版本时，Elastic Beanstalk 都会创建一个应用程序版本。如果不删除不再使用的版本，您最终会达到应用程序版本配额，并且无法创建此应用程序的新版本。

通过向应用程序应用应用程序版本生命周期策略，您可以避免达到配额。生命周期策略可要求 Elastic Beanstalk 删除应用程序的旧版本，或当应用程序的版本总数超过指定数量时删除一些应用程序版本。

Elastic Beanstalk 在您每次创建新的应用程序版本时应用应用程序生命周期策略，并在每次应用生命周期策略时删除最多 100 个版本。Elastic Beanstalk 在创建新版本后删除旧版本，并且不会将新版本计入策略中定义的最大版本数。

Elastic Beanstalk 不删除环境当前正在使用的应用程序版本，或部署到在策略触发之前十周内终止的环境的应用程序版本。

应用程序版本配额适用于区域中的所有应用程序。如果您有多个应用程序，请为每个应用程序配置适当的生命周期策略，以免达到配额。例如，如果一个区域中有 10 个应用程序并且配额为 1000 个应用程序版本，则可考虑为所有应用程序设置配额为 99 个应用程序版本的生命周期策略，或在每个应用程序中设置其他值，只要总数少于 1000 个应用程序版本即可。Elastic Beanstalk 只在应用程序版本创建成功时应用策略，因此，如果您已达到限制，则必须在创建新版本前手动删除一些版本。

默认情况下，Elastic Beanstalk 会在 Amazon S3 中保留应用程序版本源包，以防数据丢失。您可以删除源包以节省空间。

您可以通过 Elastic Beanstalk CLI 和 API 设置生命周期设置。有关详细信息，请参阅 eb appversion、CreateApplication（使用 `ResourceLifecycleConfig` 参数）和 UpdateApplicationResourceLifecycle。

在控制台中配置应用程序生命周期设置

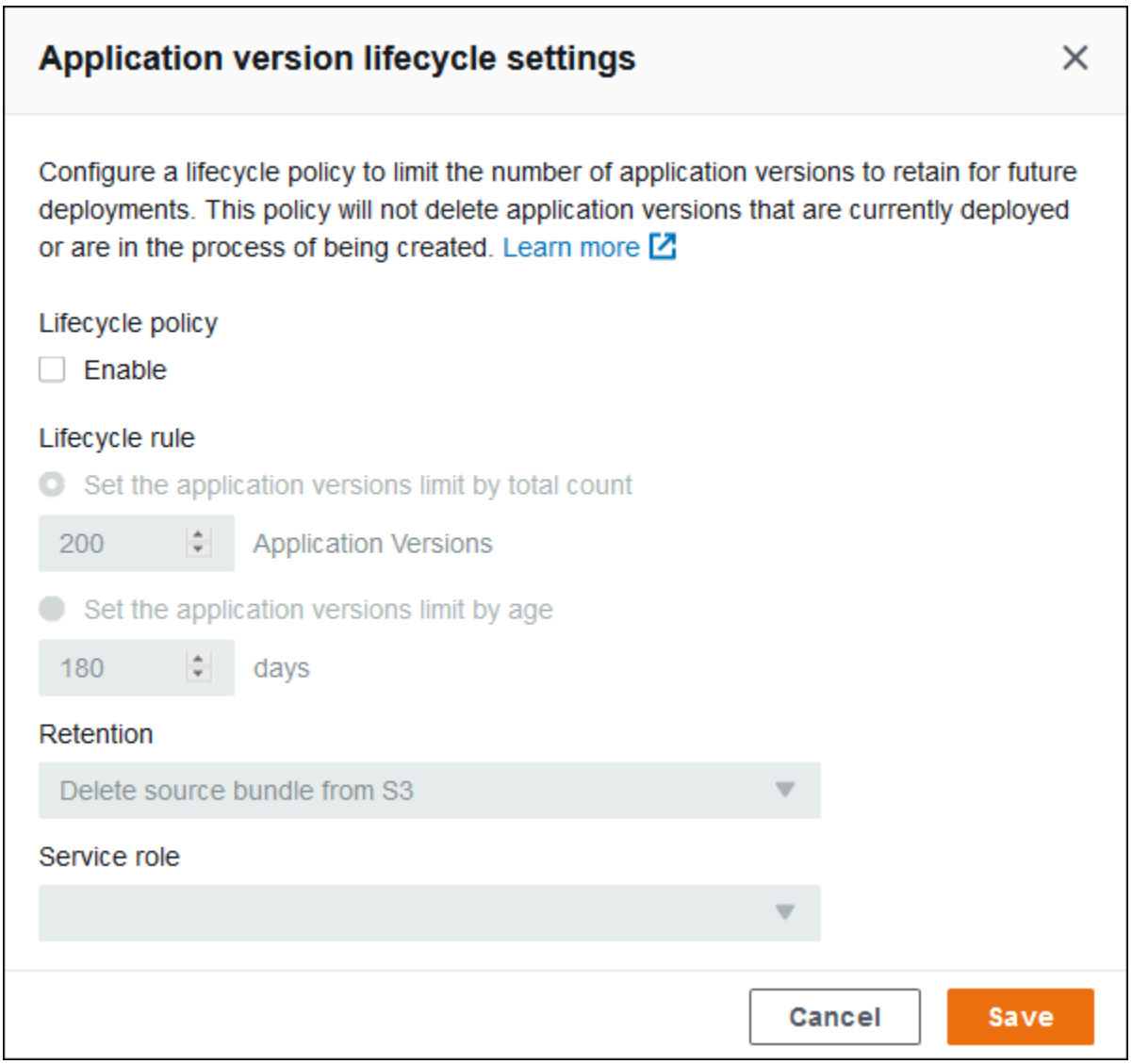
您可以在 Elastic Beanstalk 控制台中指定生命周期设置。

指定应用程序生命周期设置

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Application versions（应用程序版本）。
4. 选择 Settings。
5. 使用屏幕上的表单配置应用程序生命周期设置。
6. 选择 Save（保存）。



在设置页上，可以执行以下操作。

- 根据应用程序版本的总计数或应用程序版本的使用期限配置生命周期设置。
- 指定在删除应用程序版本时是否从 S3 删除源包。
- 指定删除应用程序版本的服务角色。要包括版本删除所需的全部权限，请选择名为 `aws-elasticbeanstalk-service-role` 的默认 Elastic Beanstalk 服务角色，或者使用 Elastic Beanstalk 托管服务策略的其他服务角色。有关更多信息，请参阅管理 Elastic Beanstalk 服务角色。

标记应用程序版本

您可以将标签应用到 AWS Elastic Beanstalk 应用程序版本。标签是与AWS资源关联的键/值对。有关 Elastic Beanstalk 资源标记、使用案例、标签键和值约束以及支持的资源类型的信息，请参阅标记 Elastic Beanstalk 应用程序资源。

您可以在创建应用程序版本时指定标签。在现有应用程序版本中，您可以添加或删除标签，以及更新现有标签的值。您最多可以为每个应用程序版本添加 50 个标签。

在创建应用程序版本期间添加标签

在使用 Elastic Beanstalk 控制台创建环境并选择上传应用程序代码版本时，您可以指定标签键和值以与新应用程序版本关联。

您还可以使用 Elastic Beanstalk 控制台上传应用程序版本，而无需立即在环境中使用该版本。您可以在上传应用程序版本时指定标签键和值。

对于 AWS CLI 或其他基于 API 的客户端，可以使用 create-application-version 命令的 `--tags` 参数添加标签。

```
$ aws elasticbeanstalk create-application-version \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --version-label v1
```

使用 EB CLI 创建或更新环境时，将根据您部署的代码创建应用程序版本。通过 EB CLI 创建应用程序版本时，没有标记应用程序版本的直接方法。有关向现有应用程序版本添加标签的信息，请参阅以下部分。

管理现有应用程序版本的标签

您可以在现有 Elastic Beanstalk 应用程序版本中添加、更新和删除标签。

使用 Elastic Beanstalk 控制台管理应用程序版本的标签

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Application versions（应用程序版本）。
4. 选择要管理的应用程序版本。
5. 选择 Actions (操作)，然后选择 Manage tags (管理标签)。
6. 使用屏幕上的表单添加、更新或删除标签。
7. 要保存更改，请选择页面底部的 Apply（应用）。

如果使用 EB CLI 更新应用程序版本，则可使用 eb tags 来添加、更新、删除或列出标签。

例如，以下命令会列出应用程序版本中的标签。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

以下命令会更新标签 mytag1 并删除标签 mytag2。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

有关选项和更多示例的完整列表，请参阅 eb tags。

对于 AWS CLI 或其他基于 API 的客户端，可使用 list-tags-for-resource 命令列出应用程序版本的标签。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

使用 update-tags-for-resource 命令可在应用程序版本中添加、更新或删除标签。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:applicationversion/my-app/my-version"
```

在 --tags-to-add 的 update-tags-for-resource 参数中指定要添加的标签和要更新的标签。添加了一个不存在的标签，更新了现有标签的值。

> **Note**
>
> 要将某些 EB CLI 和 AWS CLI 命令与 Elastic Beanstalk 应用程序版本结合使用，您需要应用程序版本的 ARN。您可以使用下面的命令检索该 ARN。
>
> ```
> $ aws elasticbeanstalk describe-application-versions --application-name my-app
>  --version-label my-version
> ```

创建应用程序源包。

使用 AWS Elastic Beanstalk 控制台部署新应用程序或应用程序版本时，需要上传源包。源包必须符合以下要求：

- 由单个 ZIP 文件或 WAR 文件组成 (您可以在 WAR 文件中包含多个 ZIP 文件)
- 不超过 500 MB
- 不包含父文件夹或顶级目录 (可包含子目录)

如果您要部署处理定期后台任务的工作线程应用程序，您的应用程序源包还必须包括一个 `cron.yaml` 文件。有关更多信息，请参阅定期任务。

如果您使用 Elastic Beanstalk Command Line Interface (EB CLI)、AWS Toolkit for Eclipse 或 AWS Toolkit for Visual Studio 部署应用程序，系统会自动正确构建 ZIP 或 WAR 文件。有关更多信息，请参阅使用 Elastic Beanstalk 命令行界面 (EB CLI)、在 Elastic Beanstalk 上创建和部署 Java 应用程序和这些区域有： AWS Toolkit for Visual Studio。

小节目录

从命令行创建源包

使用 `zip` 命令创建源包。要包含隐藏的文件和文件夹，请使用如下所示的模式。

```
~/myapp$ zip ../myapp.zip -r * .[^.]*
  adding: app.js (deflated 63%)
  adding: index.js (deflated 44%)
  adding: manual.js (deflated 64%)
  adding: package.json (deflated 40%)
  adding: restify.js (deflated 85%)
  adding: .ebextensions/ (stored 0%)
  adding: .ebextensions/xray.config (stored 0%)
```

这可确保存档中包含 Elastic Beanstalk 配置文件以及以句点开头的其他文件和文件夹。

对于 Tomcat Web 应用程序，请使用 `jar` 创建 Web 存档。

```
~/myapp$ jar -cvf myapp.war .
```

上述命令包含的隐藏文件可能会不必要地增加源包的大小。若要获得更多控制，可使用更详细的文件模式，或用 Git 创建源包。

使用 Git 创建源包

如果使用 Git 管理应用程序源代码，请使用 `git archive` 命令创建源包。

```
$ git archive -v -o myapp.zip --format=zip HEAD
```

`git archive` 仅包含 Git 中存储的文件，不包含被忽略的文件和 Git 文件。这有助于使源包尽可能小。有关更多信息，请参阅 git-archive 手册页面。

在 Mac OS X Finder 或 Windows 资源管理器中压缩文件

在 Mac OS X Finder 或 Windows 资源管理器中创建 `ZIP` 文件时，请确保压缩这些文件和子文件夹本身，而不是压缩父文件夹。

> **Note**
>
> Mac OS X 和基于 Linux 的操作系统上的图形用户界面 (GUI) 不显示名称以句点 (.) 开头的文件和文件夹。如果 `ZIP` 文件必须包含隐藏文件夹 (如 `.ebextensions`)，应使用命令行而不是 GUI 来压缩应用程序。有关在 Mac OS X 或基于 Linux 的操作系统上创建 `ZIP` 文件的命令行过程，请参阅从命令行创建源包。

Example

假设您有一个标记为 myapp 的 Python 项目文件夹，其中包含以下文件和子文件夹：

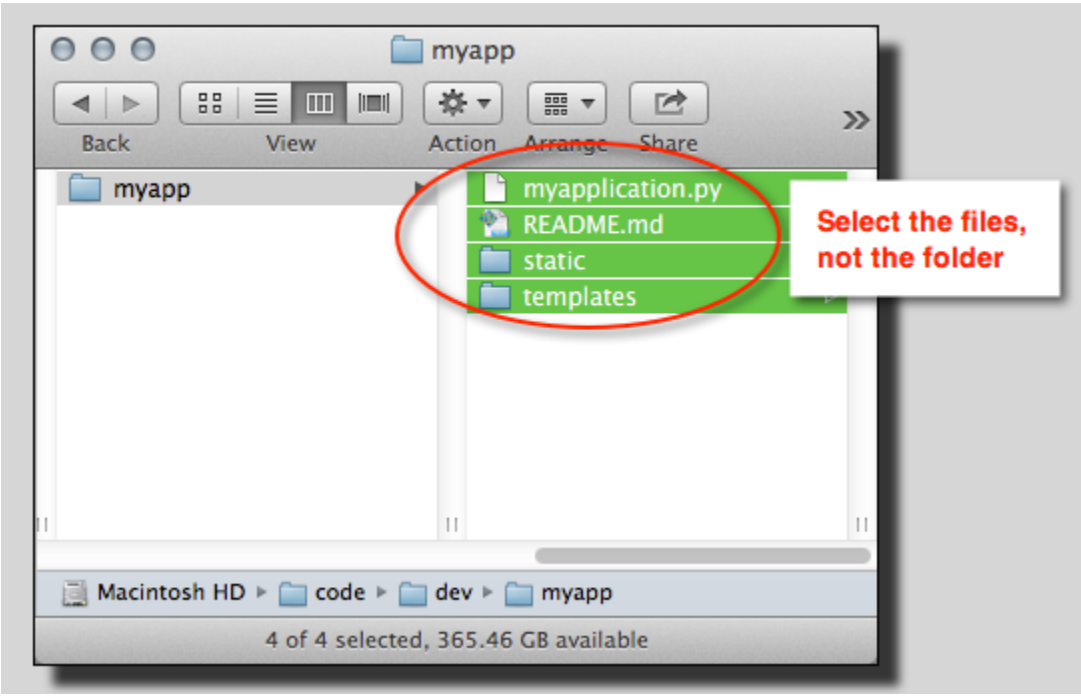
```
myapplication.py
README.md
static/
static/css
static/css/styles.css
static/img
static/img/favicon.ico
static/img/logo.png
templates/
templates/base.html
templates/index.html
```

如上面的要求列表所述，源包在压缩时必须不包含父文件夹，因此其解压缩后的结构不包含额外的顶级目录。在此示例中，解压缩文件时不应创建 myapp 文件夹 (或在命令行处，不应向文件路径添加 myapp 段)。

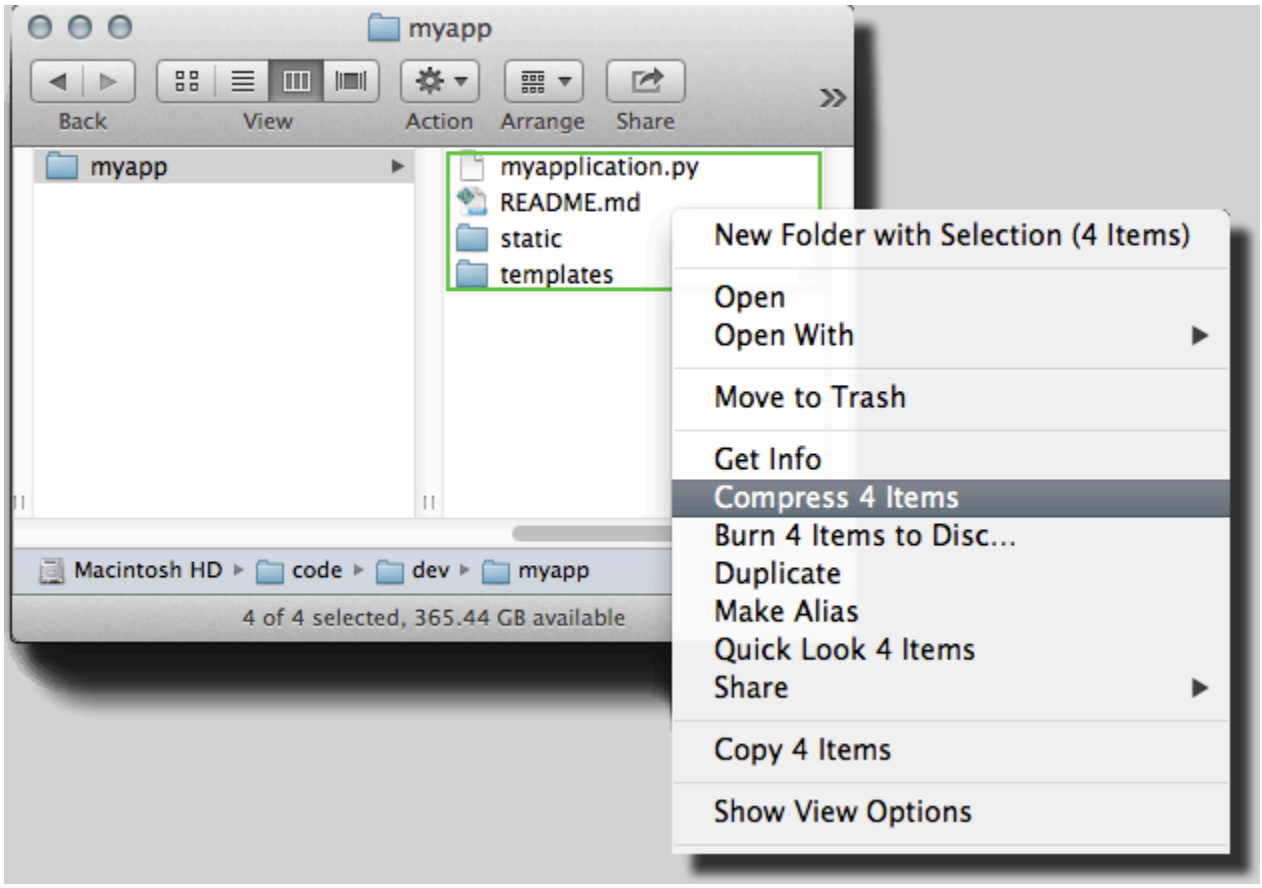
本主题均采用此示例文件结构说明如何压缩文件。

在 Mac OS X Finder 中压缩文件

1. 打开您的顶级项目文件夹，然后选择其中的所有文件和子文件夹。请勿选择顶级文件夹本身。

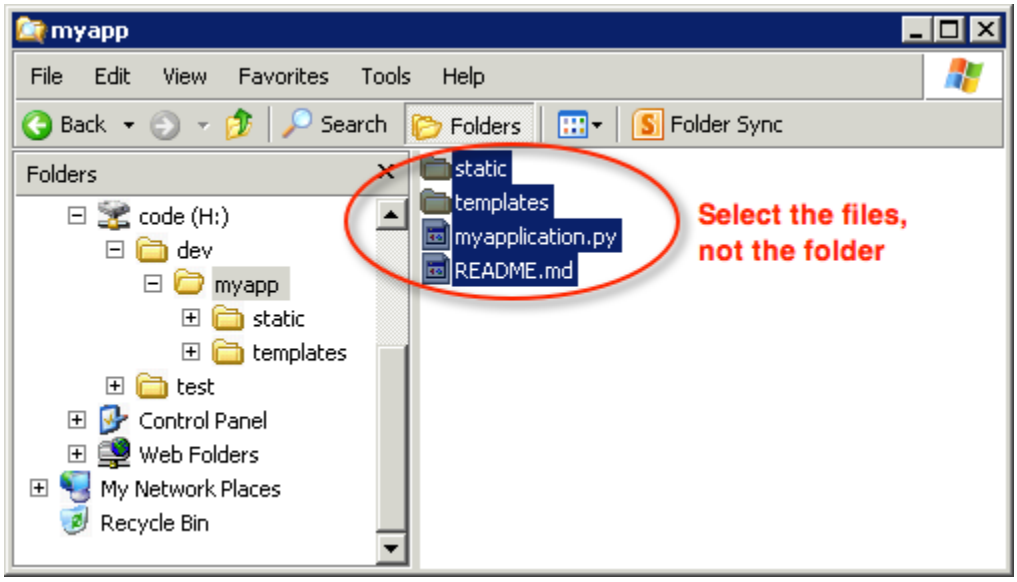


2. 右键单击所选文件，然后选择 Compress X items (压缩 X 个项目)，其中 X 是所选的文件和子文件夹数。

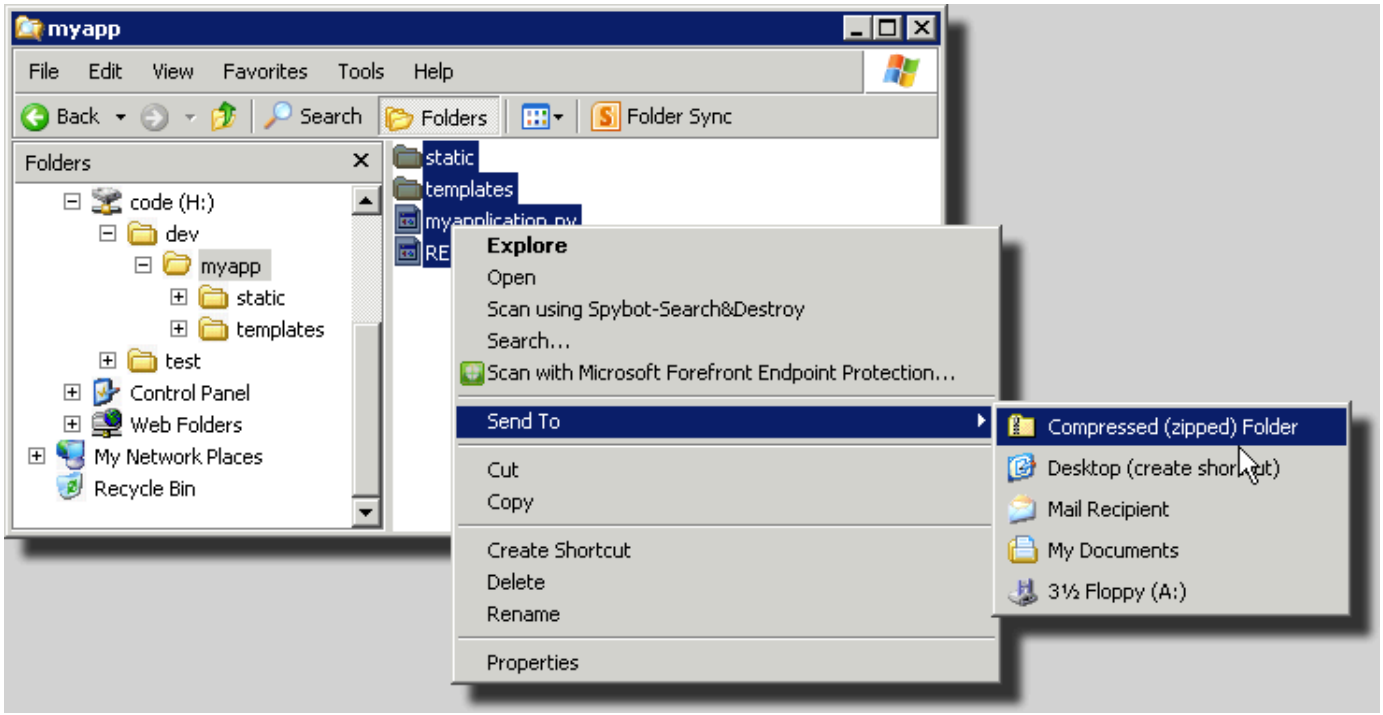


在 Windows 资源管理器中压缩文件

1. 打开您的顶级项目文件夹，然后选择其中的所有文件和子文件夹。请勿选择顶级文件夹本身。



2. 右键单击所选文件，选择 Send to (发送到)，然后选择 Compressed (zipped) folder (压缩的文件夹)。



创建 .NET 应用程序的源包

如果您使用的是 Visual Studio，则可以使用 AWS Toolkit for Visual Studio 中包含的部署工具来将 .NET 应用程序部署到 Elastic Beanstalk。有关更多信息，请参阅使用部署工具在 .NET 中部署 Elastic Beanstalk 应用程序。

如果需要手动为 .NET 应用程序创建源包，您不能直接创建一个包含项目目录的 ZIP 文件。您必须为适合部署到 Elastic Beanstalk 的项目创建一个 Web 部署包。可以使用几种方法创建部署包：

- 使用 Visual Studio 中的 Publish Web (发布 Web) 向导创建部署包。有关更多信息，请转至如何：在 Visual Studio 中创建 Web 部署程序包。

  > **Important**
  >
  > 创建 Web 部署包时，Site name (网站名称) 必须以 `Default Web Site` 开头。

- 如果是 .NET 项目，则可以使用 msbuild 命令创建部署包，如下例所示。

  > **Important**
  >
  > `DeployIisAppPath` 参数必须以 `Default Web Site` 开头。

  ```
  C:/> msbuild <web_app>.csproj /t:Package /p:DeployIisAppPath="Default Web Site"
  ```

- 如果是网站项目，则可以使用 IIS Web 部署工具创建部署包。有关更多信息，请参阅打包和还原网站。

  > **Important**
  >
  > `apphostconfig` 参数必须以 `Default Web Site` 开头。

如果您要部署多个应用程序或 ASP.NET Core 应用程序，请将 `.ebextensions` 文件夹放在源包的根目录中，与应用程序包和清单文件并排放置：

```
~/workspace/source-bundle/
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- AspNetCore101HelloWorld.zip
|-- AspNetCoreHelloWorld.zip
|-- aws-windows-deployment-manifest.json
`-- VS2015AspNetWebApiApp.zip
```

测试源包

您可能需要先本地对源包进行测试，然后再将其上传到 Elastic Beanstalk。由于 Elastic Beanstalk 基本上使用命令行提取文件，因此最好是从命令行（而不是使用 GUI 工具）进行测试。

在 Mac OS X 或 Linux 中测试文件提取

1. 打开终端窗口 (Mac OS X) 或连接到 Linux 服务器。导航到包含源包的目录。
2. 使用 `unzip` 或 `tar xf` 命令解压缩存档。
3. 确保解压缩的文件出现在与存档本身相同的文件夹中，而不是出现在新的顶级文件夹或目录中。

> **Note**
>
> 如果您使用 Mac OS X Finder 解压缩存档，则会创建新的顶级文件夹 (无论您如何构建存档本身)。为获得最佳结果，请使用命令行。

在 Windows 中测试文件提取

1. 下载或安装可用于通过命令行提取压缩文件的程序。例如，您可以从 http://stahlforce.com/dev/index.php?tool=zipunzip 下载免费的 unzip.exe 程序。
2. 如果需要，请将可执行文件复制到包含源包的目录。如果您安装了一个系统范围工具，则可以跳过此步骤。
3. 使用相应的命令解压缩存档。如果您使用步骤 1 中的链接下载了 unzip.exe，则命令为 `unzip <archive-name>`。
4. 确保解压缩的文件出现在与存档本身相同的文件夹中，而不是出现在新的顶级文件夹或目录中。

标记 Elastic Beanstalk 应用程序资源

您可以将标签应用到 AWS Elastic Beanstalk 应用程序的资源。标签是与AWS资源关联的键/值对。标签可帮助您分类资源。如果您将许多资源作为多个AWS应用程序的一部分进行管理，它们将特别有用。

以下是对 Elastic Beanstalk 资源使用标记的一些方法：

- 部署阶段 - 标识与应用程序的不同阶段关联的资源，例如开发、测试和生产。

- 成本分配 – 使用成本分配报告跟踪与各个支出账户关联的 AWS 资源的使用情况。这些报告包括已标记和未标记的资源，它们根据标签来汇总成本。有关成本分配报告如何使用标签的信息，请参阅 AWS Billing and Cost Management 用户指南中的为自定义账单报告使用成本分配标签。
- 访问控制 - 使用标签管理对请求和资源的权限。例如，只能创建和管理测试环境的用户应只能访问测试阶段的资源。有关详细信息，请参阅 使用标签控制对 Elastic Beanstalk 资源的访问。

您最多可以为每个 资源添加 50 个标签。环境略有不同：Elastic Beanstalk 会向环境添加三个默认系统标签，您无法编辑或删除这些标签。除了默认标签之外，最多可向每个环境中添加 47 个附加标签。

以下约束适用于标签键和值：

- 键和值可以包含字母、数字、空格和以下符号：`_ . : / = + - @`
- 键最多可包含 127 个字符。值最多可包含 255 个字符。

> **Note**
>
> 这些长度限制适用于 UTF-8 格式的 Unicode 字符。对于其他多字节编码，限制可能会更低。

- 键区分大小写。
- 键不能以 `aws:` 或 `elasticbeanstalk:` 开头。

标签传播到启动模板

Elastic Beanstalk 提供了便于将环境标签传播到启动模板的选项。借助此选项，可继续支持在启动模板中使用基于标签的访问控制（TBAC）。

> **Note**
>
> 正在逐步淘汰启动配置，取而代之的是启动模板。有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的启动配置。

为了防止运行停机，EC2 实例 AWS CloudFormation 不会将标签传播到现有的启动模板。如果发生需要为环境资源添加标签的应用场景，则您可以启用 Elastic Beanstalk 来为这些资源创建带有标签的启动模板。为此，请将 aws:autoscaling:launchconfiguration 命名空间中的 `LaunchTemplateTagPropagationEnabled` 选项设置为 `true`。默认值为 `false`。

以下配置文件示例可以将标签传播到启动模板。

```
option_settings:
  aws:autoscaling:launchconfiguration:
    LaunchTemplateTagPropagationEnabled: true
```

Elastic Beanstalk 只能将标签传播到以下资源的启动模板：

- EBS 卷
- EC2 实例
- EC2 网络接口
- AWS CloudFormation 启动定义资源的模板

之所以存在此限制，原因是 CloudFormation 仅允许在为特定资源创建模板时使用标签。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 TagSpecification。

> **Important**
>
> - 将现有环境的此选项值从更改 `false` 为 `true` 对于先前存在的标签可能是一项重大更改。
> - 启用此功能后，传播标签将需要更换 EC2，这可能会导致停机。您可以启用滚动更新以批量应用配置更改，并防止在更新过程中出现停机。有关更多信息，请参阅配置更改。

有关启动模板的更多信息，请参阅以下内容：

- 《Amazon EC2 Auto Scaling 用户指南》中的启动模板。
- 《AWS CloudFormation 用户指南》中的使用模板
- 《AWS CloudFormation 用户指南》中的 Elastic Beanstalk 模板片段

您可以为之添加标签的资源

以下是您可以标记的 Elastic Beanstalk 资源类型，以及为每种资源管理标签的特定主题的链接：

- 应用程序
- 环境

- 应用程序版本
- 保存的配置
- 自定义平台版本

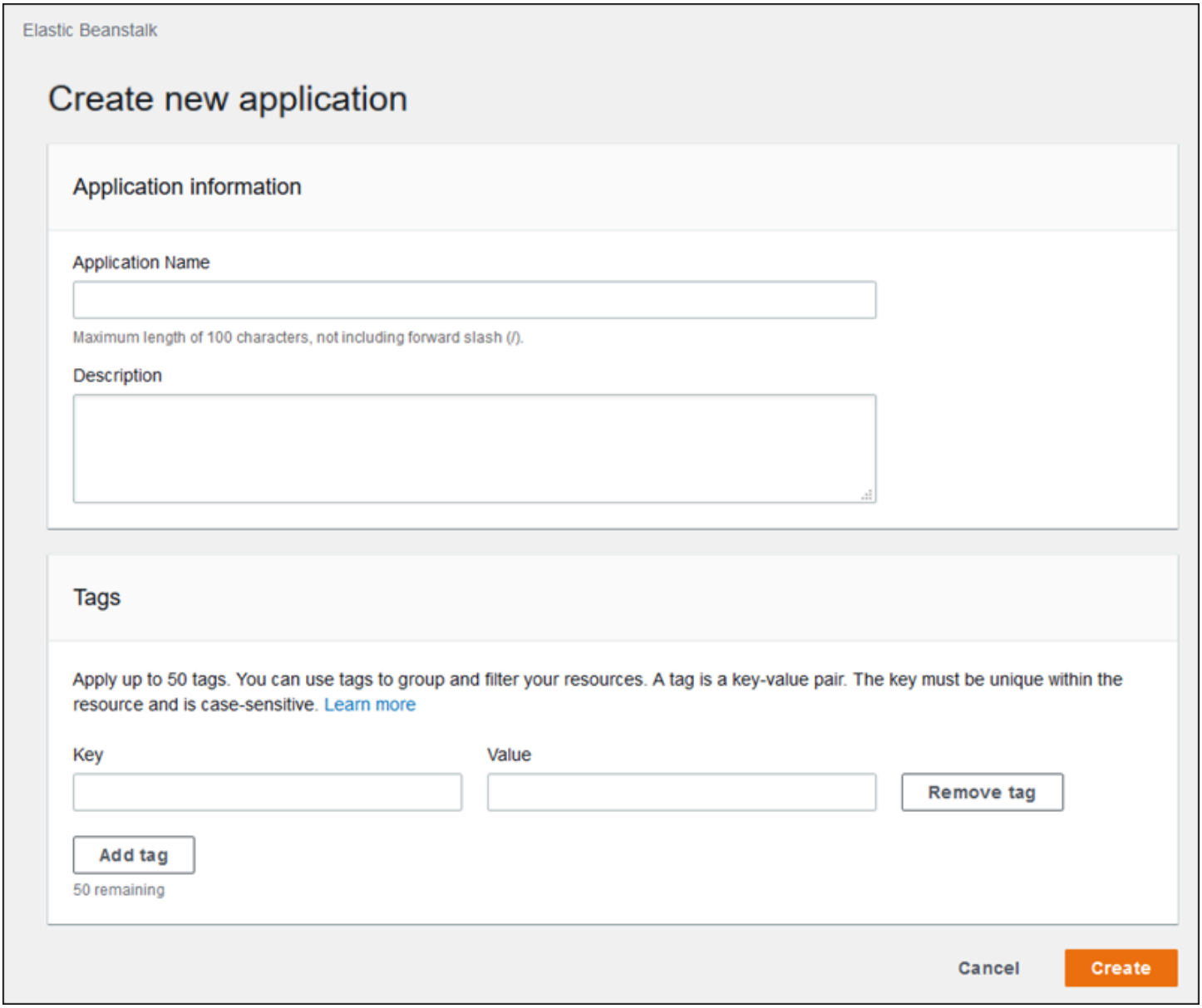
标记应用程序

您可以将标签应用到 AWS Elastic Beanstalk 应用程序。标签是与AWS资源关联的键/值对。有关 Elastic Beanstalk 资源标记、使用案例、标签键和值约束以及支持的资源类型的信息，请参阅标记 Elastic Beanstalk 应用程序资源。

您可以在创建应用程序时指定标签。在现有应用程序中，您可以添加或删除标签，以及更新现有标签的值。您最多可以为每个应用程序添加 50 个标签。

在创建应用程序期间添加标签

在使用 Elastic Beanstalk 控制台创建应用程序时，您可以在 Create New Application (创建新应用程序) 对话框中指定标签键和值。



如果使用 EB CLI 创建应用程序，则可以使用 eb init 的 `--tags` 选项添加标签。

```
~/workspace/my-app$ eb init --tags mytag1=value1,mytag2=value2
```

对于 AWS CLI 或其他基于 API 的客户端，可以使用 create-application 命令的 `--tags` 参数添加标签。

```
$ aws elasticbeanstalk create-application \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --version-label v1
```

管理现有应用程序的标签

可以在现有的 Elastic Beanstalk 应用程序中添加、更新和删除标签。

在 Elastic Beanstalk 控制台中管理应用程序的标签

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 选择 Actions (操作)，然后选择 Manage tags (管理标签)。
4. 使用屏幕上的表单添加、更新或删除标签。
5. 要保存更改，请选择页面底部的 Apply（应用）。

如果使用 EB CLI 更新应用程序，则可使用 eb tags 来添加、更新、删除或列出标签。

例如，以下命令会列出应用程序中的标签。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-app"
```

以下命令会更新标签 mytag1 并删除标签 mytag2。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-app"
```

有关选项和更多示例的完整列表，请参阅 eb tags。

对于 AWS CLI 或其他基于 API 的客户端，可使用 list-tags-for-resource 命令列出应用程序的标签。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-app"
```

使用 update-tags-for-resource 命令可在应用程序中添加、更新或删除标签。

```
$ aws elasticbeanstalk update-tags-for-resource \
```

```
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:application/my-
app"
```

在 `--tags-to-add` 的 update-tags-for-resource 参数中指定要添加的标签和要更新的标签。添加了一个不存在的标签，更新了现有标签的值。

> **Note**
>
> 要将某些 EB CLI 和 AWS CLI 命令与 Elastic Beanstalk 应用程序结合使用，您需要应用程序的 ARN。您可以使用下面的命令检索该 ARN。
>
> ```
> $ aws elasticbeanstalk describe-applications --application-names my-app
> ```

管理环境

AWS Elastic Beanstalk 可以轻松创建新的应用程序环境。您可以创建和管理用于开发、测试和生产使用的单独环境，还可以向任意环境部署任意版本的应用程序。环境可以是长时间运行的或临时的。在终止环境时，可以保存其配置以便将来重新创建环境。

在开发应用程序时，会经常进行部署，可能出于不同目的部署到多个不同环境。使用 Elastic Beanstalk 可以配置执行部署的方式。您可以同时向环境中的所有实例进行部署，也可以采用滚动部署方式分批部署。

配置更改是与部署分别处理的，有其自己的范围。例如，如果更改运行您的应用程序的 EC2 实例的类型，则必须替换所有实例。另一方面，如果修改环境的负载均衡器配置，可以就地进行更改，无需中断服务或降低容量。也可以通过滚动配置更新分批应用可修改环境中实例的配置更改。

> **Note**
>
> 仅使用 Elastic Beanstalk 修改环境中的资源。如果使用其他服务的控制台、CLI 命令或开发工具包修改资源，Elastic Beanstalk 将无法准确监控这些资源的状态，并且您将无法保存配置或可靠地重新创建环境。带外更改也会在更新或终止环境时导致问题。

在启动环境时，可选择平台版本。我们会使用新的平台版本定期更新平台，以提供性能改进和新功能。您随时可以将您的环境更新为最新平台版本。

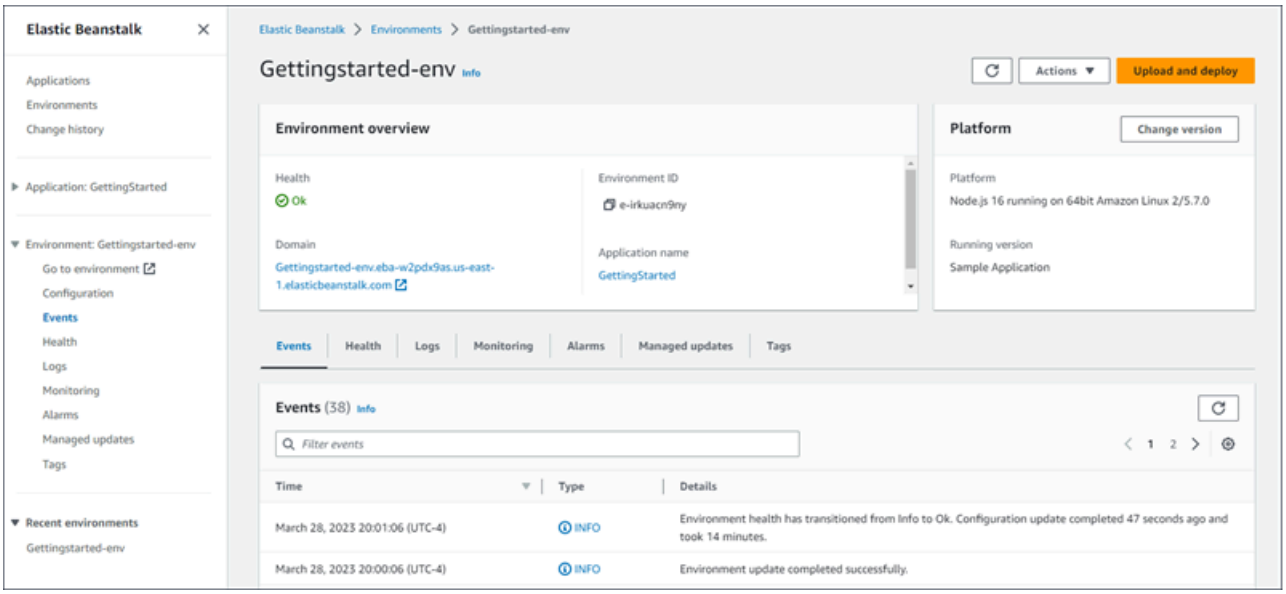
随着应用程序复杂性的增长，您可以将其拆分为多个组件，每个组件在一个单独的环境中运行。对于长时间运行的工作负载，您可以启动工作线程环境来处理来自 Amazon Simple Queue Service (Amazon SQS) 队列的作业。

主题

- Elastic Beanstalk 工作线程环境
- 在 Elastic Beanstalk 环境之间创建链接

使用 Elastic Beanstalk 环境管理控制台

Elastic Beanstalk 控制台提供了一个 Environment overview（环境概述）页面，供您管理每个 AWS Elastic Beanstalk 环境。在 Environment overview（环境概述）页面中，您可以管理环境的配置并执行常见操作。这些操作包括重新启动环境中运行的 Web 服务器、克隆环境或从头开始重建环境。



访问环境管理控制台

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

您将看到 Environment overview（环境概述）页面。控制台的导航窗格显示环境所属的应用程序的名称（以及相关的应用程序管理页面）和环境名称（以及环境管理页面）。

主题

- 环境概述

- 环境操作
- 事件
- 健康
- 日志
- 监控
- 告警
- 托管更新
- 标签
- 配置

环境概述

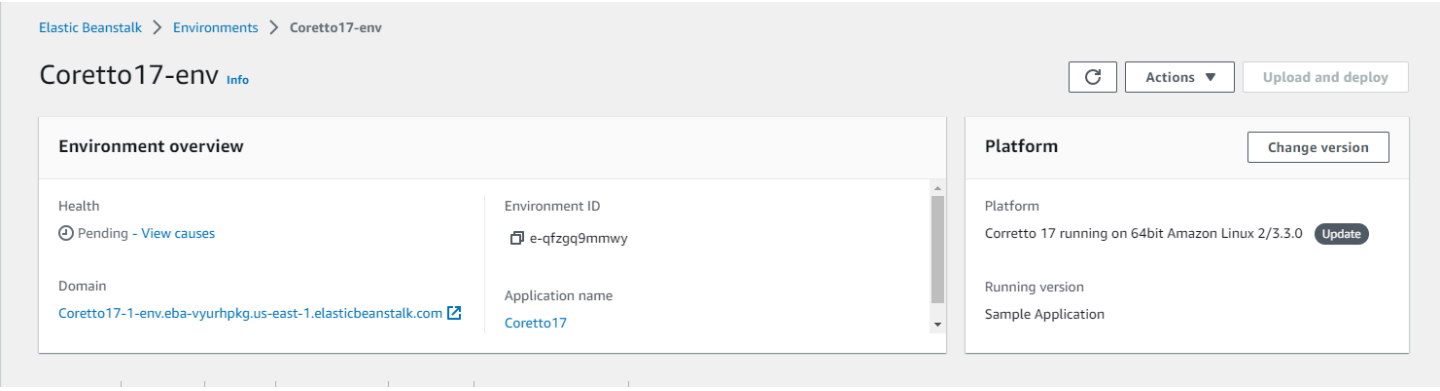
要查看 Environment overview（环境概述）页面，请在导航窗格上选择环境名称（如果它是当前环境）。或者，从 Applications（应用程序）页面或 Environments（环境）页面上的主环境列表导航到环境。

环境概述页面上的顶部窗格显示有关您的环境的顶级信息。这包括其名称、URL、当前运行状况，以及当前部署的应用程序版本的名称和运行应用程序的平台版本。在概述窗格下方，您可以看到最新的环境事件。

选择 Refresh（刷新）可更新显示的信息。概述页面包含以下信息和选项。

健康

环境的总体运行状况。如果您的环境运行状况恶化，则环境运行状况旁边会显示查看原因链接。选择此链接可查看运行状况选项卡，其中包含更多详细信息。



Domain

环境的域或 URL 位于 Environment overview（环境概述）页面的上半部分，环境的 Health（运行状况）下方。这是环境正在运行的 Web 应用程序的 URL。

环境 ID

环境 ID。这是创建环境时生成的内部 ID。

应用程序名称

在您的环境中部署并运行的应用程序的名称。

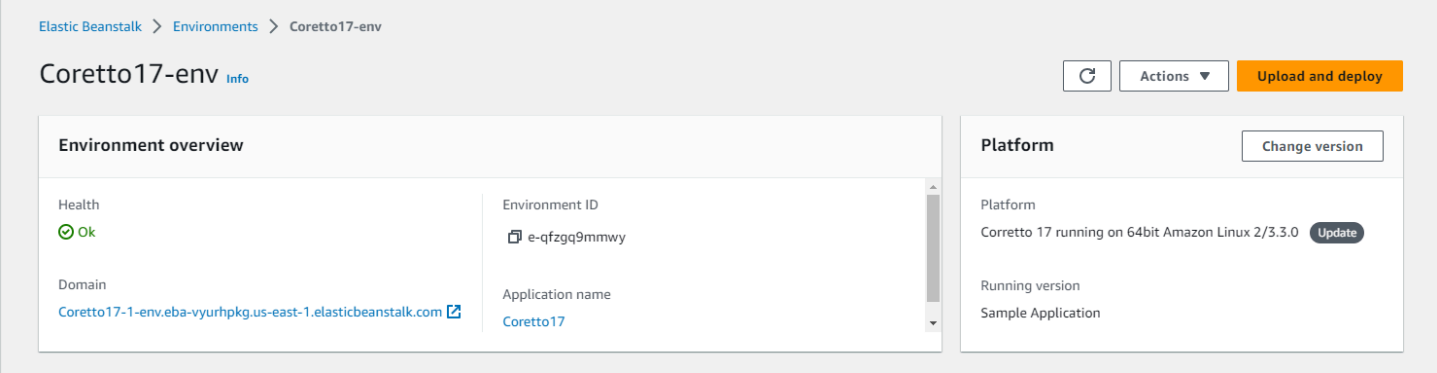
运行版本

在环境中部署并运行的应用程序版本的名称。选择 Upload and deploy（上传和部署）可上传源包并将其部署到环境。此选项创建新应用程序版本。

平台

在环境上运行的平台版本的名称。通常，这包括架构、操作系统（OS）、语言和应用程序服务器（统称为平台分支）以及特定平台版本号的组合。

如果您的平台版本不是最新版本，则在平台部分的旁边会显示一个状态标签。更新标签表明，尽管支持该平台版本，但有更新的版本可用。平台版本也可能被标记为已弃用或已停用。选择更改版本将您的平台分支更新到更新的版本。有关平台版本状态的更多信息，请参阅 Elastic Beanstalk 平台词汇表 中的平台分支部分。



环境概述选项卡

该页面下半部分显示的选项卡包含有关环境的更多详细信息，并提供对其他功能的访问：

- Events (事件) - 显示来自 Elastic Beanstalk 服务以及此环境使用其资源的其他服务的信息或错误消息。
- Health (运行状况) - 显示有关运行应用程序的 Amazon EC2 实例的状态和详细运行状况信息。

- Logs（日志）– 从您的环境中的 Amazon EC2 检索和下载日志。您可以检索完整的日志或最近的活动。检索到的日志在 15 分钟内可用。
- Monitoring (监控) - 显示环境的统计数据（如平均延迟和 CPU 使用率）。
- Alarms（警报）– 显示您为环境指标配置的警报。您可以在此页面上添加、修改或删除警报。
- Managed updates（托管更新）– 显示有关即将到来和已完成的托管平台更新以及实例替换的信息。
- Tags (标签) - 显示环境标签，并允许您对其进行管理。标签是应用于您的环境的密钥值对。

> **Note**
>
> 控制台左侧的导航窗格列出了与选项卡同名的链接。选择这些链接中的任何一个都将显示相应选项卡的内容。

环境操作

环境概述页面包含可用于对环境执行常见操作的 Actions（操作）菜单。此菜单显示在环境标题右侧的 Create a new environment（创建新环境）选项旁。

> **Note**
>
> 一些操作只有在特定条件下可用，除非满足适当条件，否则将保持禁用状态。

加载配置

加载之前保存的配置。配置会保存到您的应用程序，并可由任意关联的环境加载。如果您对环境的配置进行了更改，则可以加载保存的配置以撤消这些更改。您还可以加载从运行同一应用程序的其他环境保存的配置，以在环境之间传播配置更改。

保存配置

将环境的当前配置保存到您的应用程序。对环境的配置进行更改之前，请保存当前配置，以便您在需要时回滚。当您启动新环境时，还可以应用保存的配置。

交换环境域 (URL)

将当前环境的别名记录与新环境交换。别名记录交换之后，使用环境 URL 指向应用程序的所有流量将指向新环境。当您准备好部署应用程序的新版本时，您可以在新版本下启动单独的环境。当您的新环境

准备好开始接收请求时，执行别名记录交换可以开始将流量路由到新环境。这样做不会中断您的服务。有关更多信息，请参阅使用 Elastic Beanstalk 进行蓝/绿部署。

克隆环境

启动与您当前正在运行的环境具有相同配置的新环境。

使用最新平台进行克隆

使用最新版本的正在使用的 Elastic Beanstalk 平台，克隆您的当前环境。只有在当前环境的平台具有较新可用版本时，此选项才可用。

中止当前操作

停止正在进行的环境更新。停止操作会导致环境中的一些实例与另一些实例处于不同的状态，具体取决于操作的进度。此选项仅在您的环境正在更新时可用。

重启应用程序服务器

重新启动在您的环境实例上运行的 Web 服务器。此选项不终止或重新启动任何AWS资源。如果您的环境在响应一些错误请求时表现出了奇怪的行为，重新启动应用程序可能会暂时恢复功能，同时您可以找出根本原因来排除故障。

重建环境

终止正在运行环境中的所有资源，然后使用相同设置构建新环境。此操作需要几分钟时间，相当于从头开始部署新环境所需的时间。重建期间，任何运行在您的环境数据层中的 Amazon RDS 实例都将删除。如果您需要数据，请创建快照。您可以在 RDS 控制台中手动创建快照，或者配置数据层的删除策略，以在删除实例之前自动创建快照。这是创建数据层时的默认设置。

终止环境

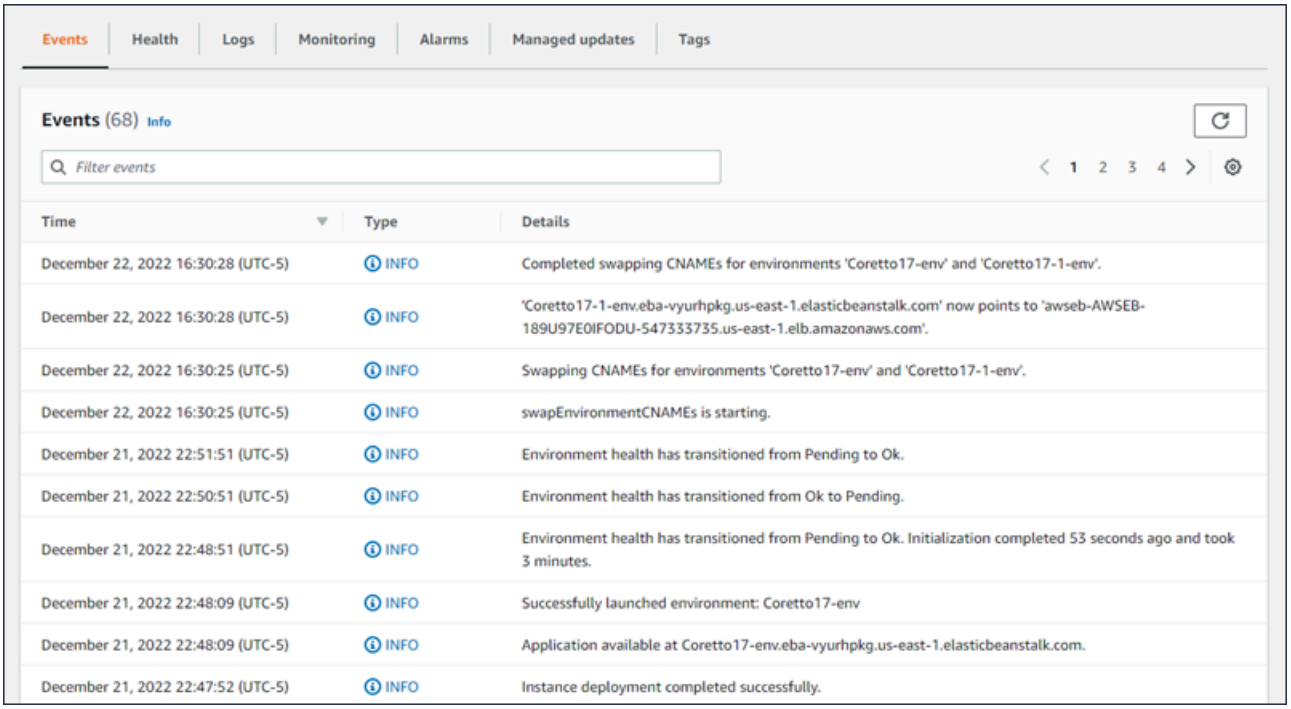
终止正在运行环境中的所有资源，然后从应用程序中删除环境。如果您有在数据层中运行的 RDS 实例并且需要保留数据，请确保数据库删除策略被设置为 `Snapshot` 或者 `Retain`。有关更多信息，请参阅本指南的配置环境章节中的数据库生命周期。

恢复环境

如果环境在过去一小时内终止，您可以从此页面恢复环境。如果超过一小时，您可以从应用程序概览页面恢复环境。

事件

Events（事件）选项卡显示您的环境的事件流。在您与环境交互时，以及作为结果创建或修改了环境的任意资源时，Elastic Beanstalk 会输出事件消息。

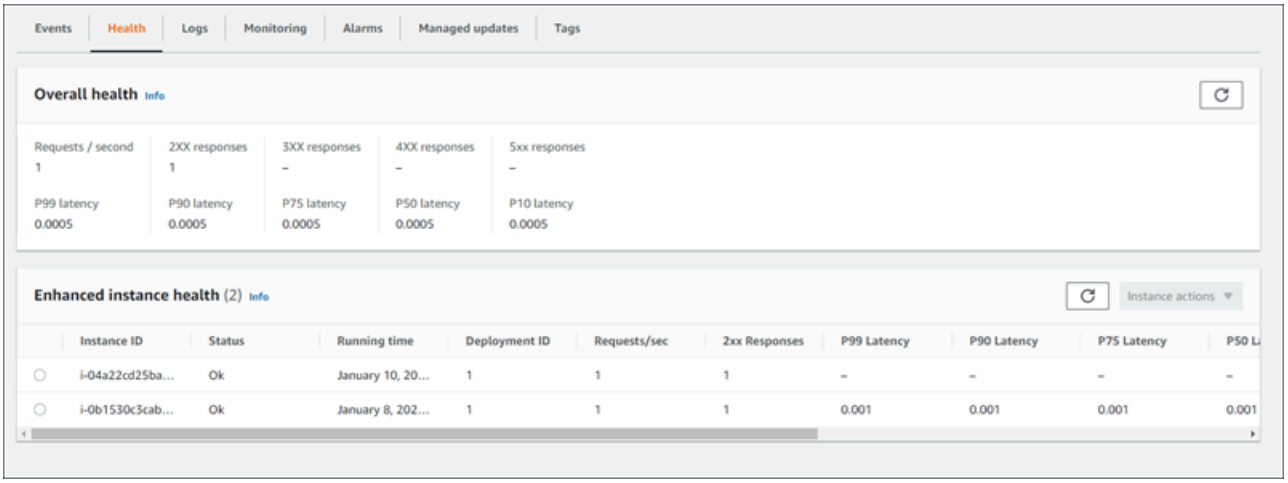


有关更多信息，请参阅查看 Elastic Beanstalk 环境的事件流。

健康

如果启用了增强型运行状况监控，此页面会显示实例的实时运行状况信息。Overall health（总体运行状况）窗格将运行状况数据显示为环境中所有实例的平均值。Enhanced instance health（增强型实例运行状况）窗格显示环境中每个实例的实时运行状况信息。增强型运行状况监控使得 Elastic Beanstalk 可以密切监控环境中的资源，从而更加准确地评估应用程序的运行状况。

启用了增强型运行状况监视时，此页显示有关您的环境中实例所处理请求的信息，以及来自操作系统的指标，包括延迟、负载和 CPU 利用率。

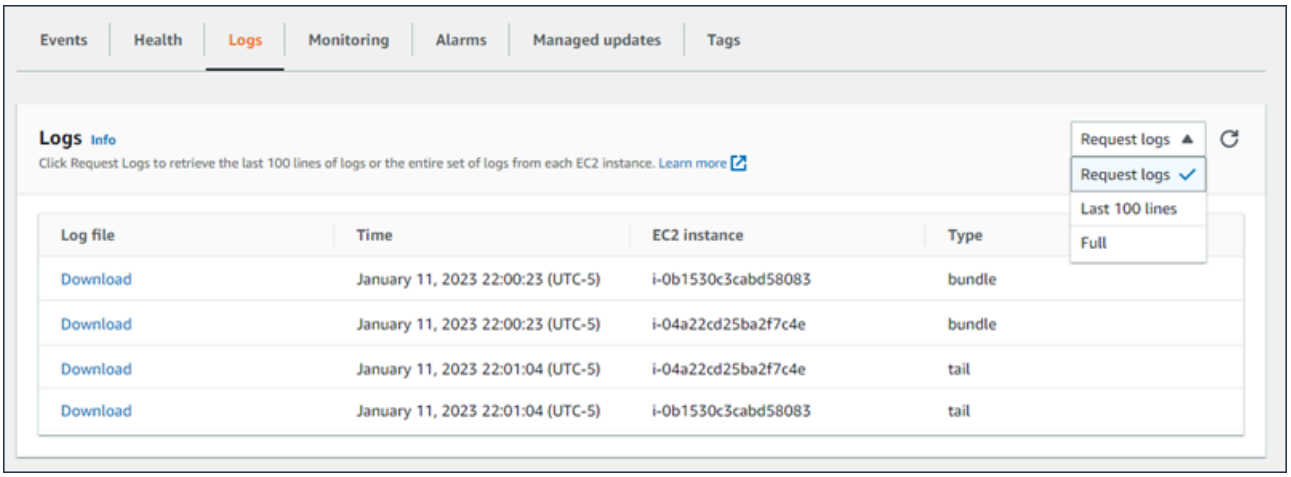


有关更多信息，请参阅增强型运行状况报告和监控。

日志

使用 Logs（日志）页可以检索您环境中 EC2 实例的日志。在请求日志时，Elastic Beanstalk 发送命令到实例，然后实例将日志上传到 Amazon S3 中您的 Elastic Beanstalk 存储桶。当您在此页面上请求日志时，Elastic Beanstalk 会在 15 分钟后自动从 Amazon S3 中删除它们。

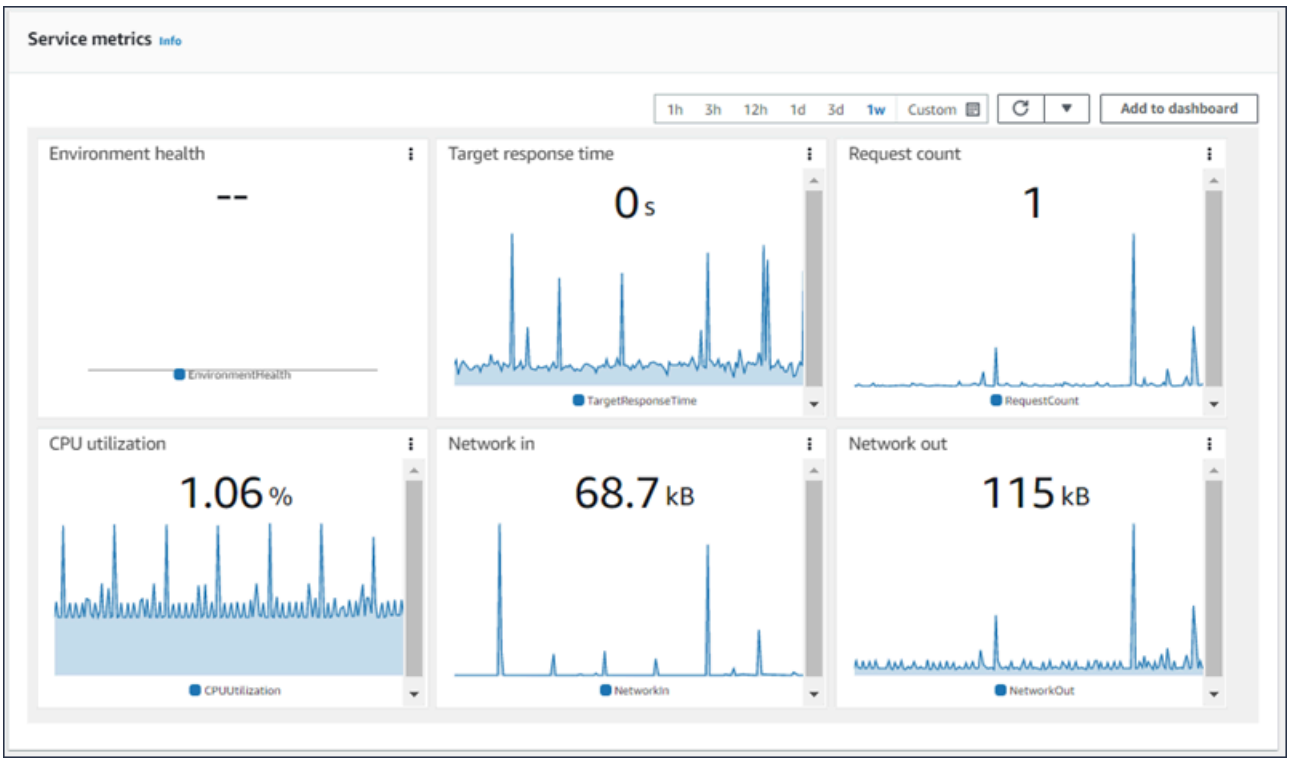
您还可以将环境实例配置为将日志上传到 Amazon S3，以便在本地轮换之后永久存储这些日志。



有关更多信息，请参阅查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志。

监控

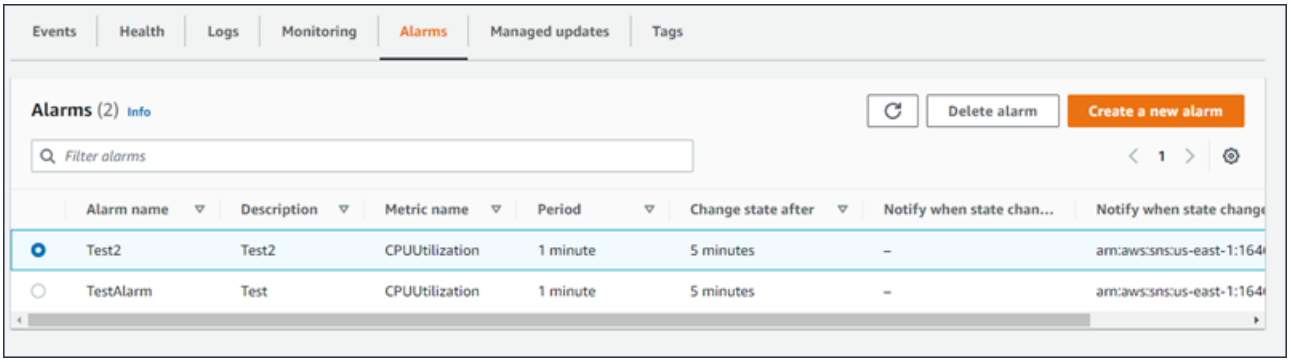
Monitoring（监控）页面显示您的环境的运行状况信息概述。这包括 Elastic Load Balancing 和 Amazon EC2 提供的默认指标集，以及显示环境运行状况随时间变化的图表。



有关更多信息，请参阅在AWS管理控制台中监控环境运行状况。

告警

Existing alarms（现有警报）页面显示您为环境配置的任意警报的相关信息。您可以使用此页上的选项创建或删除警报。



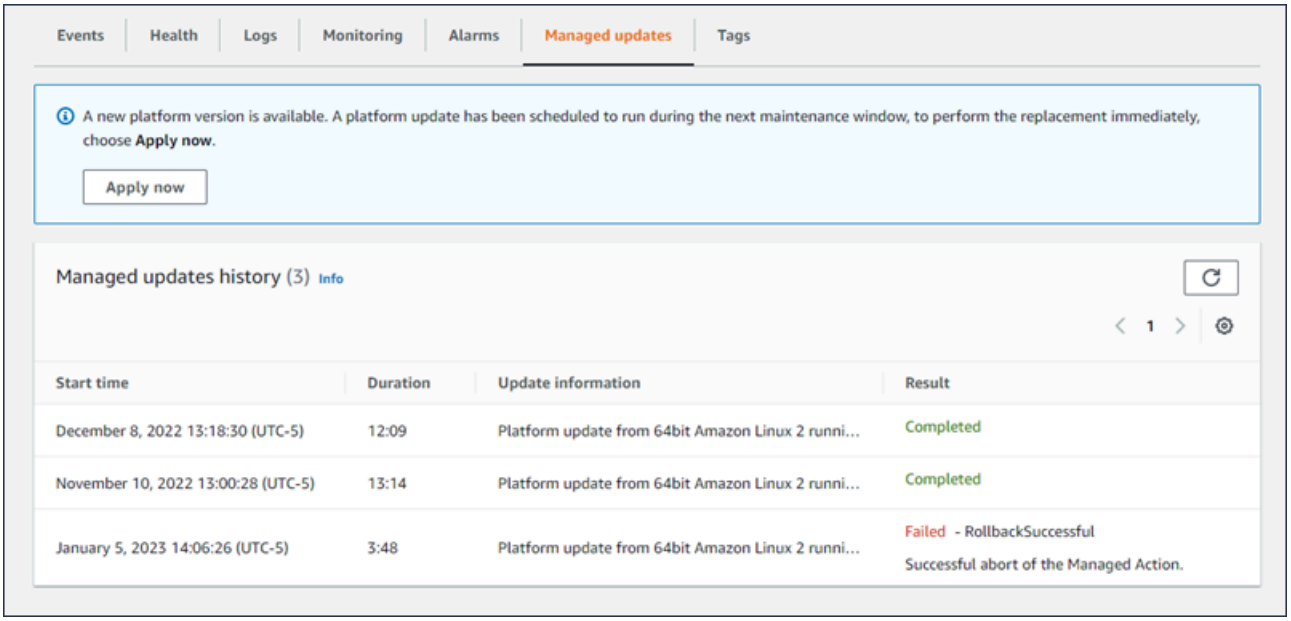
有关更多信息，请参阅管理警报。

托管更新

Managed updates overview（托管更新概述）页面显示有关即将发布的和已完成的托管平台更新以及实例替换的信息。

您可以使用托管更新功能将环境配置为在您选择的每周维护时段内自动更新到最新平台版本。在不同的平台版本之间，您可选择在维护时段内让环境替换所有其 Amazon EC2 实例。这可以减少在应用程序长时间运行时发生的问题。

有关更多信息，请参阅托管平台更新。

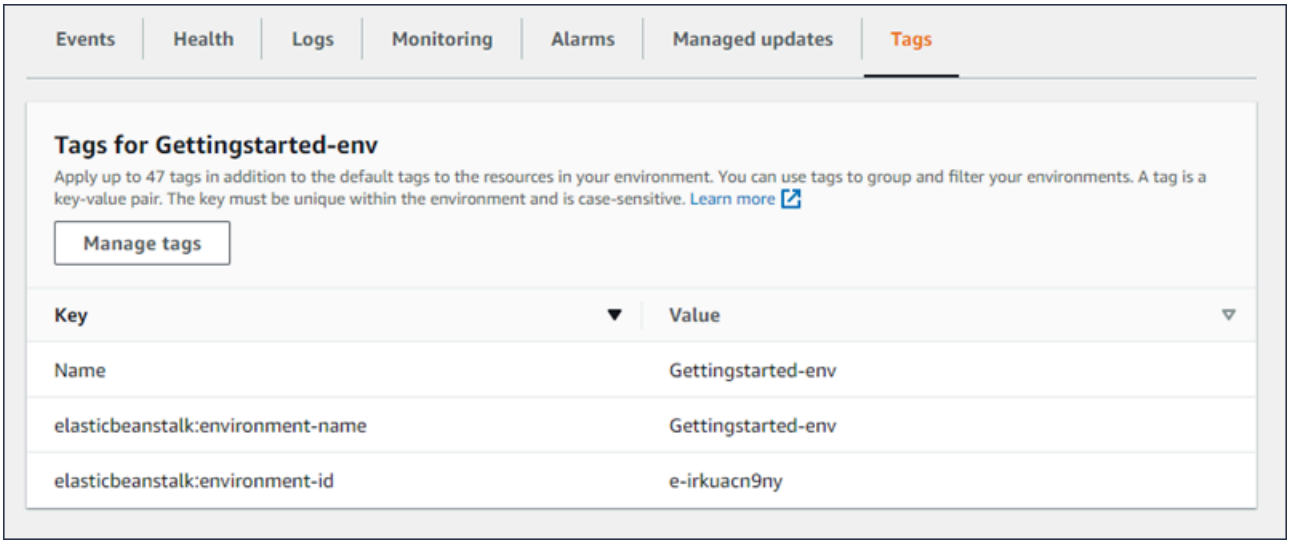


有关更多信息，请参阅托管平台更新。

标签

标签页面显示 Elastic Beanstalk 在您创建环境时应用于环境的标签以及您添加的任何标签。可以添加、编辑和删除自定义标签。您无法编辑或删除 Elastic Beanstalk 应用的标签。

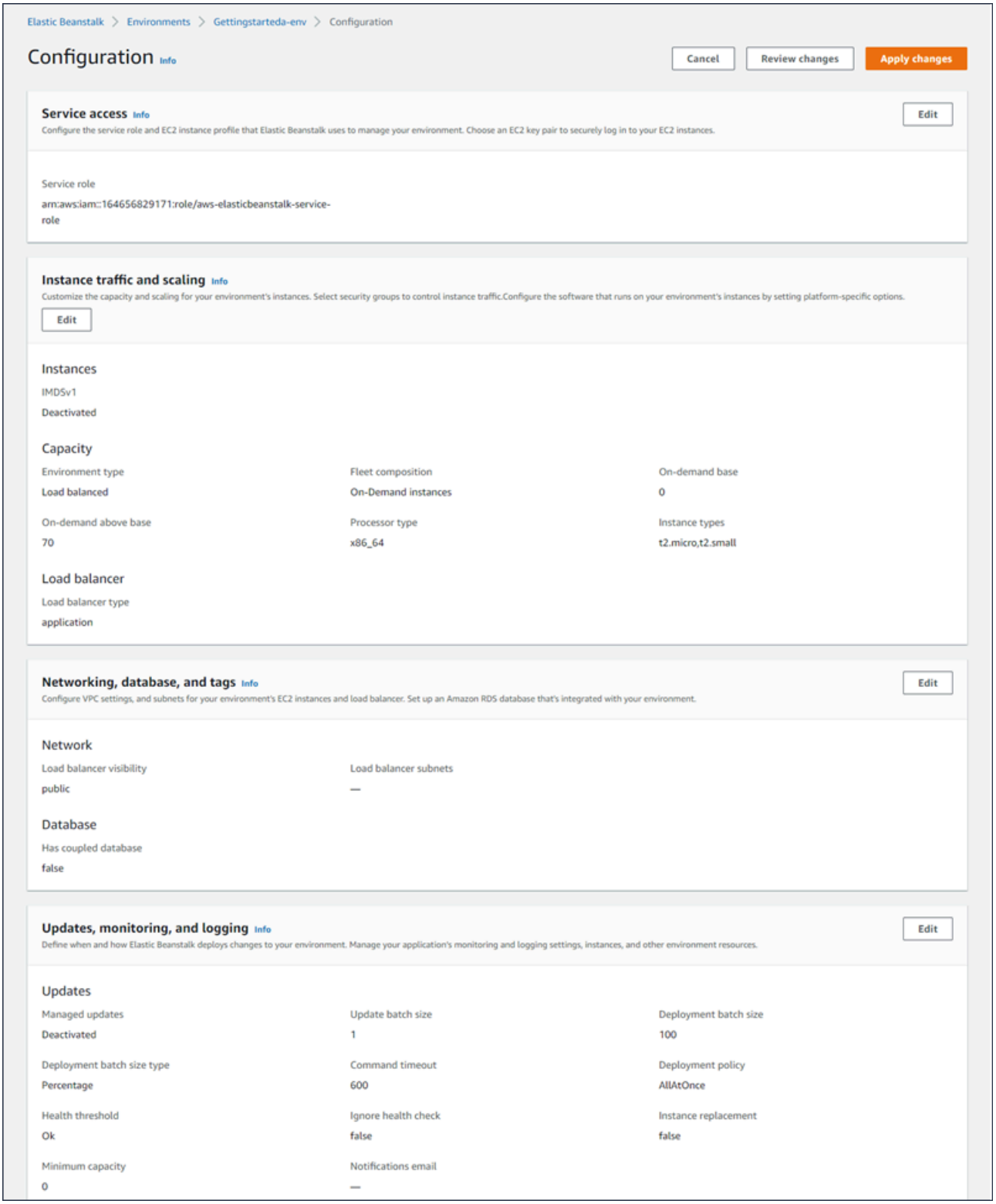
环境标签将应用到 Elastic Beanstalk 为了支持您的应用程序而创建的所有资源。



有关更多信息，请参阅在 Elastic Beanstalk 环境中标记资源。

配置

Configuration（配置）页面显示您的环境当前的配置及其资源，包括 Amazon EC2 实例、负载均衡器、通知以及运行状况监控设置。在部署期间使用此页上的设置自定义您环境的行为、启用附加功能，以及修改实例类型和您在创建环境期间选择的其他设置。



有关更多信息，请参阅[配置 Elastic Beanstalk 环境](#)。

创建 Elastic Beanstalk 环境

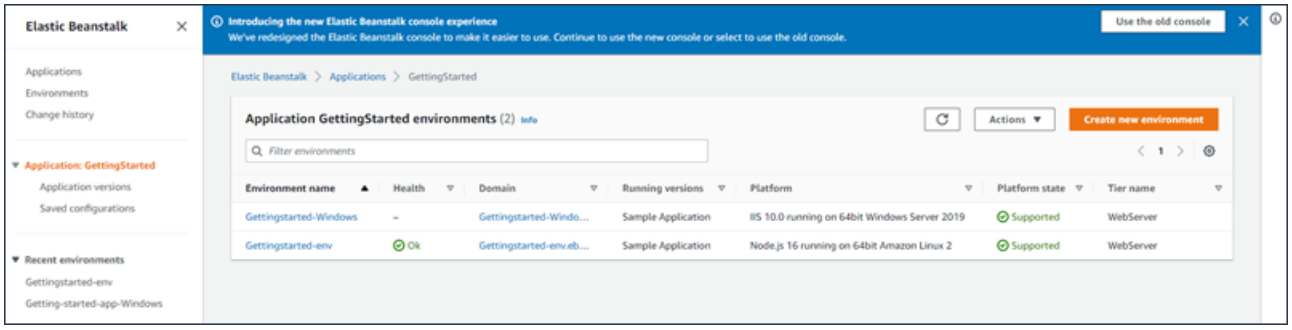
AWS Elastic Beanstalk 环境 是运行应用程序版本的 AWS 资源的集合。当需要运行应用程序的多个版本时，您可部署多个环境。例如，您可能会有开发、集成和生产环境。

在以下过程中，将启动运行默认应用程序的新环境。这些步骤进行了简化，以便使用默认选项值快速启动并运行您的环境。有关可用来配置资源（Elastic Beanstalk 代表您部署）的许多选项的详细说明，请参阅创建新环境向导。

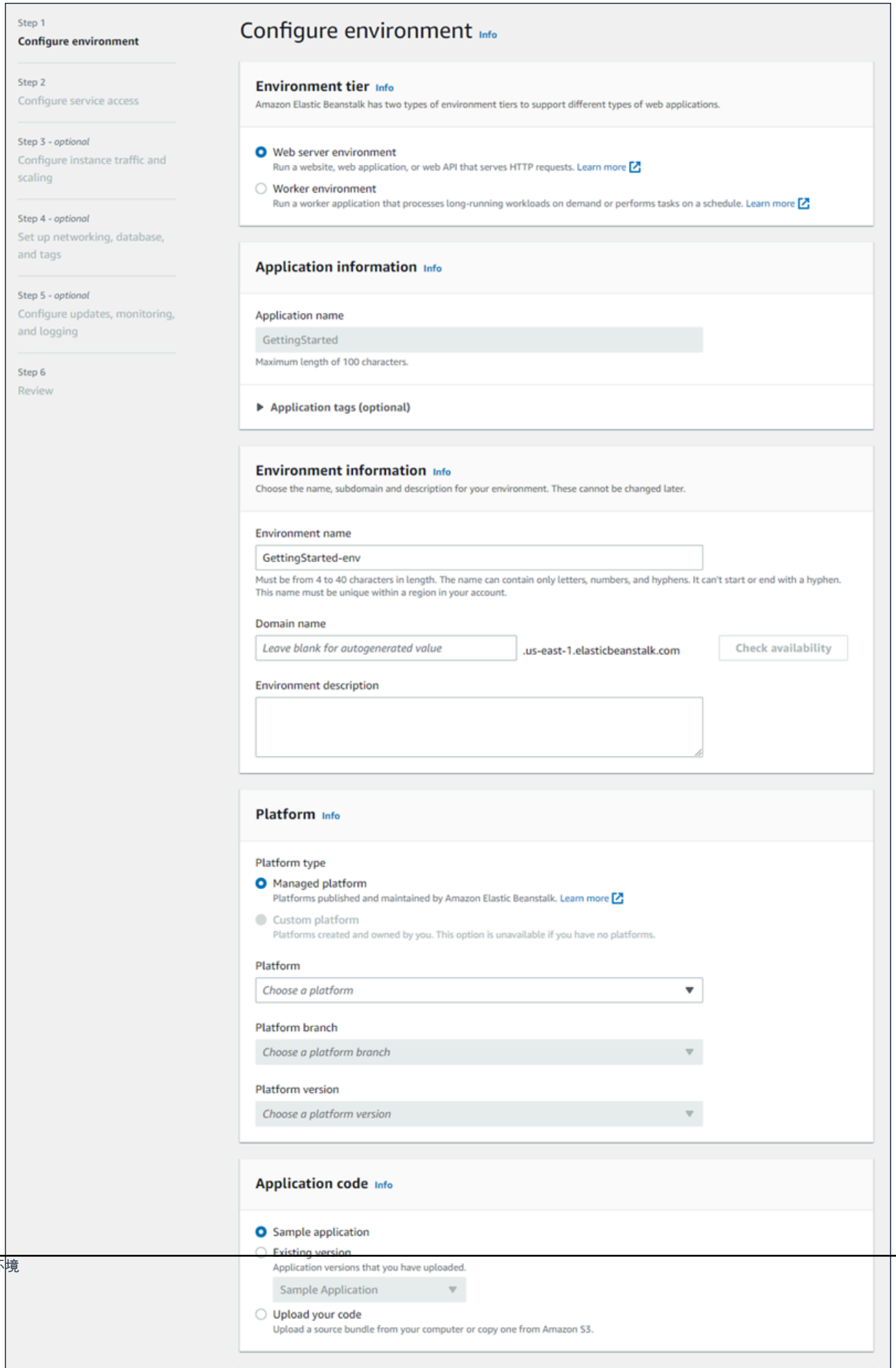
> **注意**
>
> - 有关使用 EB CLI 创建和管理环境的说明，请参阅使用 EB CLI 管理 Elastic Beanstalk 环境。
> - 创建环境需要在 Elastic Beanstalk 中完全访问托管式策略的权限。有关更多信息，请参阅 Elastic Beanstalk 用户策略。

使用示例应用程序启动环境 (控制台)

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择应用程序，然后在列表中选择现有应用程序的名称或创建一个。
3. 在应用程序概述页面上，选择 Create new environment（创建新环境）。



这将启动 Create environment（创建环境）向导。该向导提供了一组创建新环境的步骤。


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Configure instance traffic and scaling
Step 4 - optional
Set up networking, database, and tags
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure environment Info
Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more
Application information Info
Application name
GettingStarted
Maximum length of 100 characters.
Application tags (optional)
Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.
Environment name
GettingStarted-env
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.
Domain name
Leave blank for autogenerated value
.us-east-1.elasticbeanstalk.com
Check availability
Environment description
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Choose a platform
Platform branch
Choose a platform branch
Platform version
Choose a platform version
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Sample Application
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

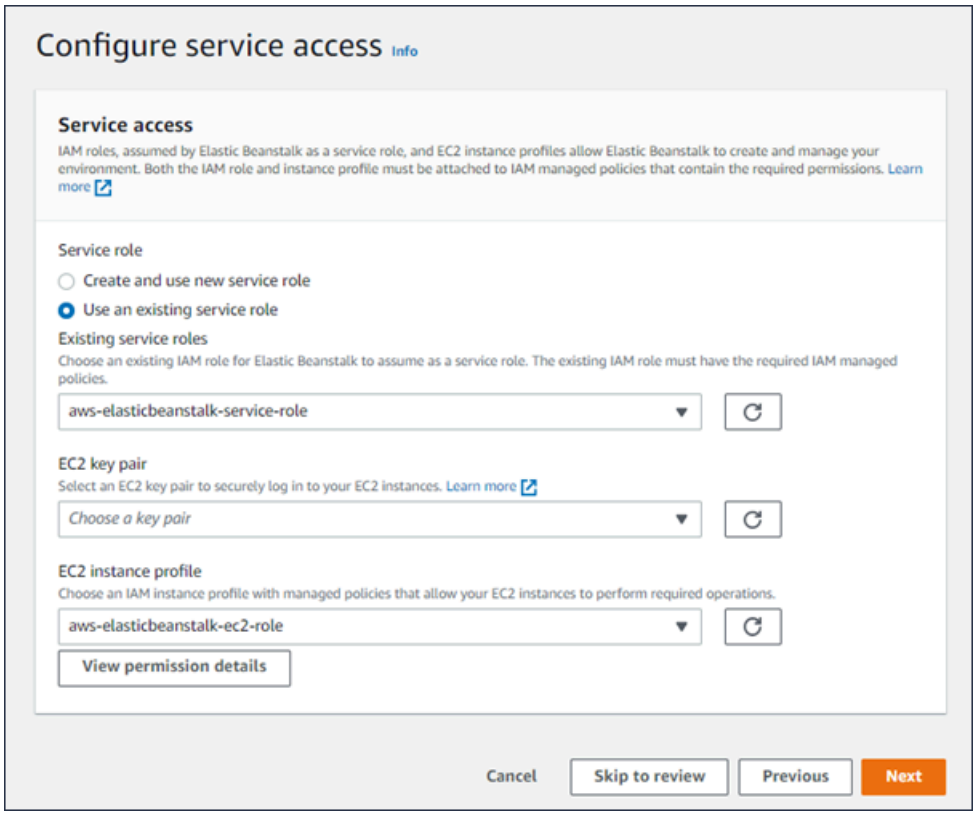
4. 对于环境层，选择 Web server environment（Web 服务器环境）或 Worker environment（工作线程环境）环境层。环境的层创建后无法更改。

> **Note**
>
> .NET on Windows Server 平台不支持工作线程环境层。

5. 对于平台，选择与应用程序使用的语言匹配的平台和平台分支。

> **Note**
>
> Elastic Beanstalk 支持列出的大多数平台的多个版本。默认情况下，此控制台将为您选择的平台和平台分支选择推荐版本。如果您的应用程序需要其他版本，您可以在此处选择该版本。有关支持的平台版本的信息，请参阅the section called “支持的平台”。

6. 对于应用程序代码，选择示例应用程序。
7. 对于 Configuration presets（配置预设），选择 Single instance（单一实例）。
8. 选择 Next（下一步）。
9. 这时将显示配置服务访问权限页面。



10. 对于服务角色，请选择使用现有的服务角色。
11. 下面我们将重点介绍 EC2 实例配置文件下拉列表。此下拉列表中显示的值可能因您的账户之前是否创建过新环境而异。

    根据列表中显示的值，选择以下选项中的一个。

    - 如果在下拉列表中显示有 `aws-elasticbeanstalk-ec2-role`，请从 EC2 实例配置文件下拉列表中将其选中。
    - 如果列表中显示的是其他值，并且这是您环境的默认 EC2 实例配置文件，请从 EC2 实例配置文件下拉列表中选中该值。
    - 如果 EC2 实例配置文件下拉列表未显示任何可供选择的值，请按下面为 EC2 实例配置文件创建 IAM 角色的过程操作。

      完成为 EC2 实例配置文件创建 IAM 角色中的步骤，以创建一个之后为 EC2 实例配置文件选择的 IAM 角色。然后返回此步骤。

      现在您已创建了一个 IAM 角色并刷新了列表，该角色将在下拉列表中显示为一个选项。从 EC2 实例配置文件下拉列表中选中您刚刚创建的 IAM 角色。
12. 在 Configure service access（配置服务访问）页面上选择 Skip to Review（跳至审核）。

    这样做将选择此步骤的默认值，并跳过可选步骤。
13. Review（审核）页面将显示所有选择的摘要。

    要进一步自定义您的环境，请在包含要配置的任何项目的步骤旁边选择 Edit（编辑）。只能在创建环境期间设置下列选项：

    - 环境名称
    - 域名
    - 平台版本
    - 处理器
    - VPC
    - 套餐

    可在环境创建后更改下列设置，但它们需要配置新实例或其他资源并且应用更改可能需要很长的时间：

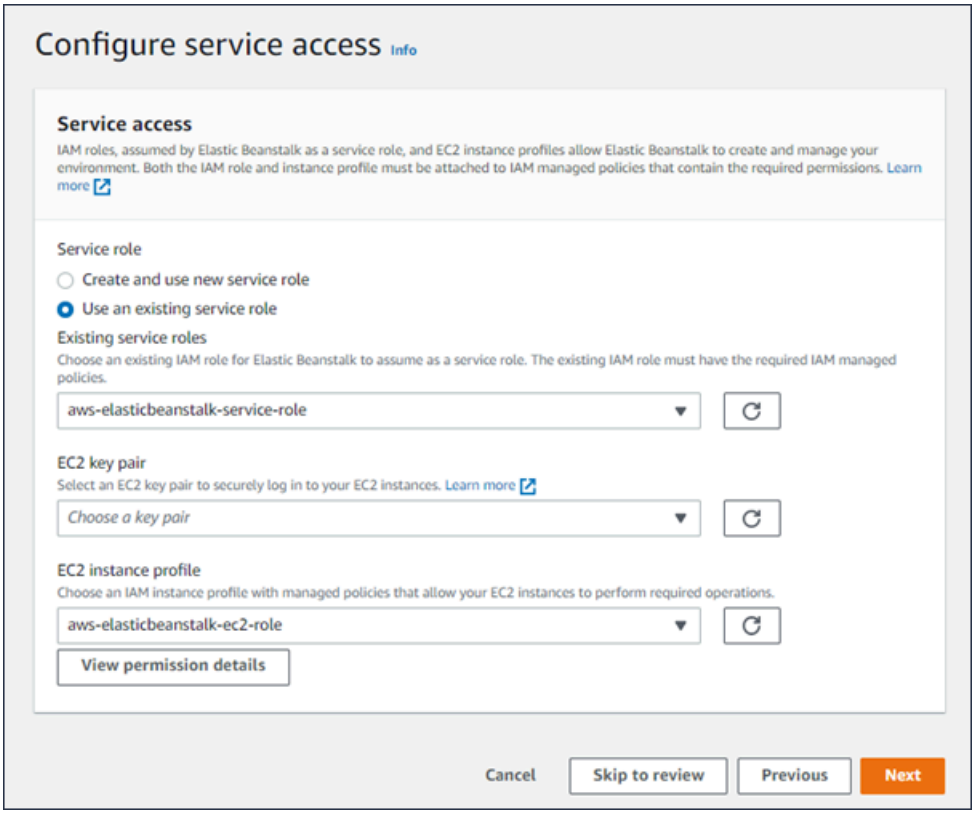
    - 实例类型、根卷、密钥对和 AWS Identity and Access Management (IAM) 角色

- 内部 Amazon RDS 数据库
- 负载均衡器

有关所有可用设置的详细信息，请参阅创建新环境向导。

14. 选择页面底部的 Submit（提交）以初始化新环境的创建。

为 EC2 实例配置文件创建 IAM 角色



创建用于 EC2 实例配置文件选择的 IAM 角色

1. 选择查看权限详细信息。这将在 EC2 实例配置文件下拉列表下显示。

   这时会显示一个名为查看实例配置文件权限的模态窗口。此窗口将列出您需要附加到所创建的新 EC2 实例配置文件的托管式配置文件。此外还提供了一个用于启动 IAM 控制台的链接。

2. 选择窗口顶部显示的 IAM 控制台链接。
3. 请在 IAM 控制台的导航窗格中，选择 Roles（角色）。
4. 选择 Create role（创建角色）。
5. 在可信实体类型下，选择 AWS 服务。

6. 在 Use case（使用案例）下，选择 EC2。
7. 选择 Next（下一步）。
8. 附加适当的托管式策略。滚动查看实例配置文件权限模式窗口，以查看托管式策略。这些策略还将在此处列出：
   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 选择 Next（下一步）。
10. 输入角色的名称。
11. （可选）将标签添加到角色。
12. 选择 Create role（创建角色）。
13. 返回已打开的 Elastic Beanstalk 控制台窗口。
14. 关闭查看实例配置文件权限模态窗口。

> **Important**
> 不要关闭显示 Elastic Beanstalk 控制台的浏览器页面。

15. 选择 EC2 实例配置文件下拉列表旁边的  （刷新）。

    这将刷新下拉列表，以确保您刚刚创建的角色会在下拉列表中显示。

当 Elastic Beanstalk 创建环境时，您将被重定向到 Elastic Beanstalk 控制台。当环境运行状况变绿后，选择环境名称旁的 URL 可查看运行的应用程序。除非您将环境配置为使用带内部负载均衡器的自定义 VPC，否则此 URL 一般可通过 Internet 访问。

主题

- 创建新环境向导
- 克隆 Elastic Beanstalk 环境
- 终止 Elastic Beanstalk 环境
- 使用 AWS CLI 创建 Elastic Beanstalk 环境

- 使用 API 创建 Elastic Beanstalk 环境
- 构建 Launch Now URL
- 创建和更新 Elastic Beanstalk 环境组

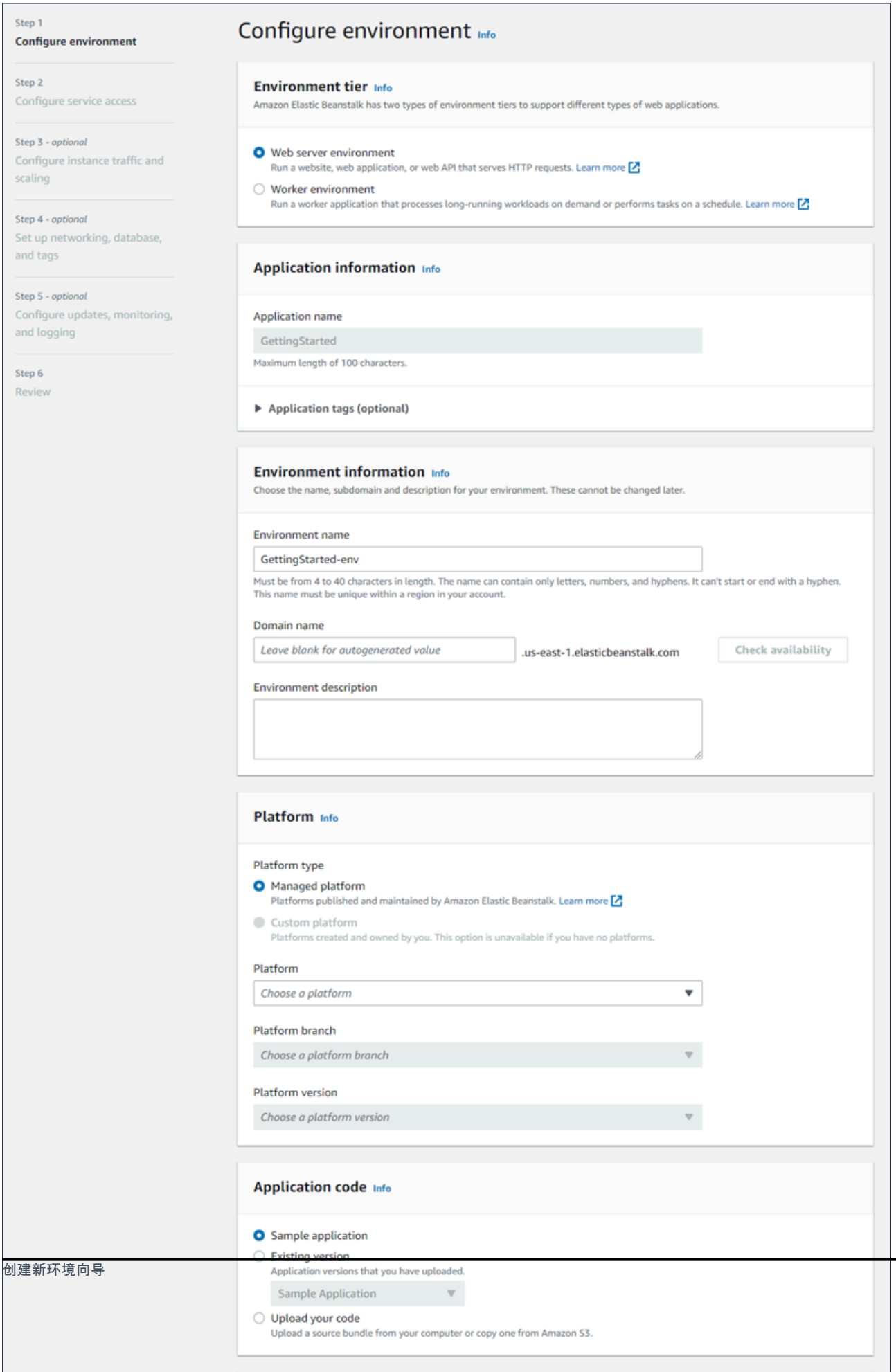
创建新环境向导

在 创建 Elastic Beanstalk 环境 中，我们介绍了如何打开 Create environment（创建环境）向导并快速创建环境。选择 Create environment（创建环境）可启动具有默认环境名称、自动生成的域、示例应用程序代码和建议设置的环境。

本主题介绍了 Create environment（创建环境）向导以及可以使用该向导配置要创建的环境的所有方式。

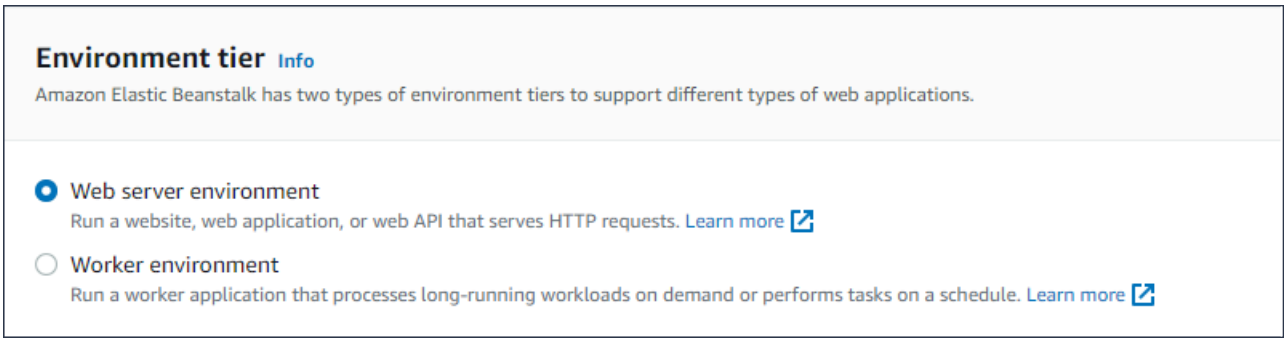
向导页面

Create environment（创建环境）向导提供了一组创建新环境的步骤。


Step 1
Configure environment
Step 2
Configure service access
Step 3 - optional
Configure instance traffic and scaling
Step 4 - optional
Set up networking, database, and tags
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Configure environment Info
Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.
Web server environment
Run a website, web application, or web API that serves HTTP requests. Learn more
Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. Learn more
Application information Info
Application name
GettingStarted
Maximum length of 100 characters.
Application tags (optional)
Environment information Info
Choose the name, subdomain and description for your environment. These cannot be changed later.
Environment name
GettingStarted-env
Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.
Domain name
Leave blank for autogenerated value
.us-east-1.elasticbeanstalk.com
Check availability
Environment description
Platform Info
Platform type
Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. Learn more
Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.
Platform
Choose a platform
Platform branch
Choose a platform branch
Platform version
Choose a platform version
Application code Info
Sample application
Existing version
Application versions that you have uploaded.
Sample Application
Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

环境层

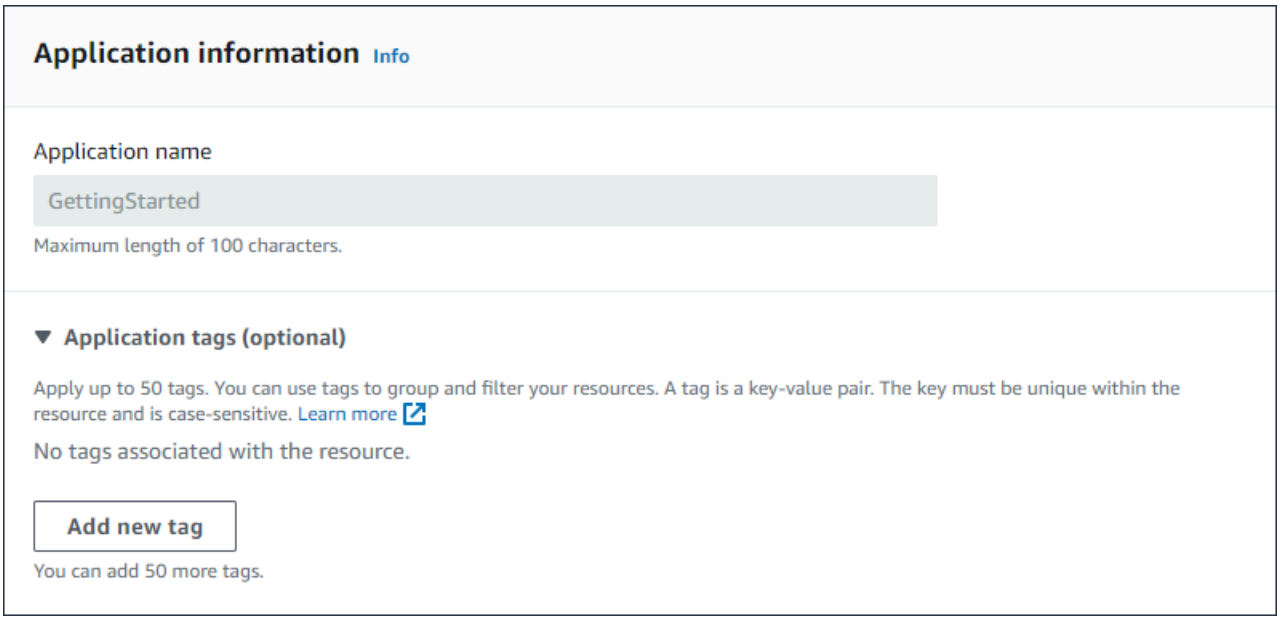
对于环境层，选择 Web server environment（Web 服务器环境）或 Worker environment（工作线程环境）环境层。环境的层创建后无法更改。



> **Note**
>
> .NET on Windows Server 平台不支持工作线程环境层。

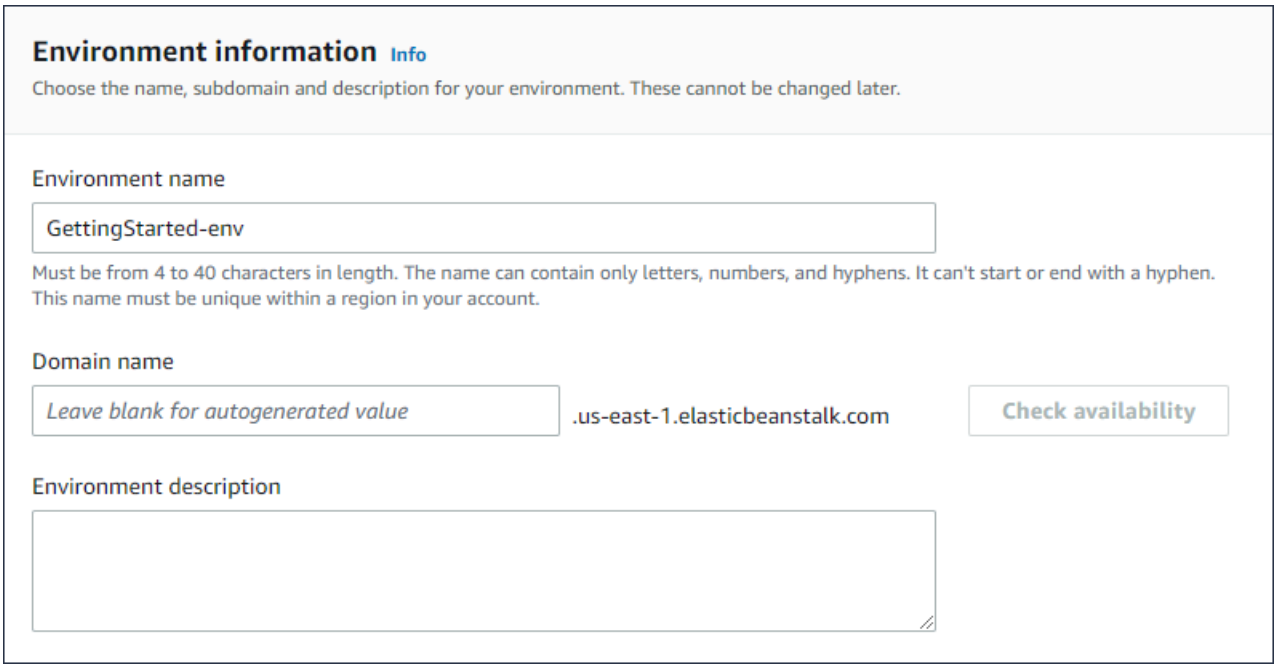
应用程序信息

如果您通过从 Application overview（应用程序概述）页面中选择 Create new environment（创建新环境）来启动向导，则 Application name（应用程序名称）将预先填充。否则，请输入应用程序名称。（可选）添加应用程序标签。



环境信息

设置环境的名称和域，并为您的环境创建描述。请注意，在创建环境之后无法更改这些环境设置。



- 名称 – 输入环境的名称。该窗体提供生成的名称。
- 域 –（Web 服务器环境）为您的环境输入唯一的域名。默认名称是环境的名称。您可以输入其他域名。Elastic Beanstalk 使用此名称为环境创建唯一的 CNAME。要检查所需的域名是否可用，请选择 Check Availability（检查可用性）。
- 描述 – 输入对此环境的描述。

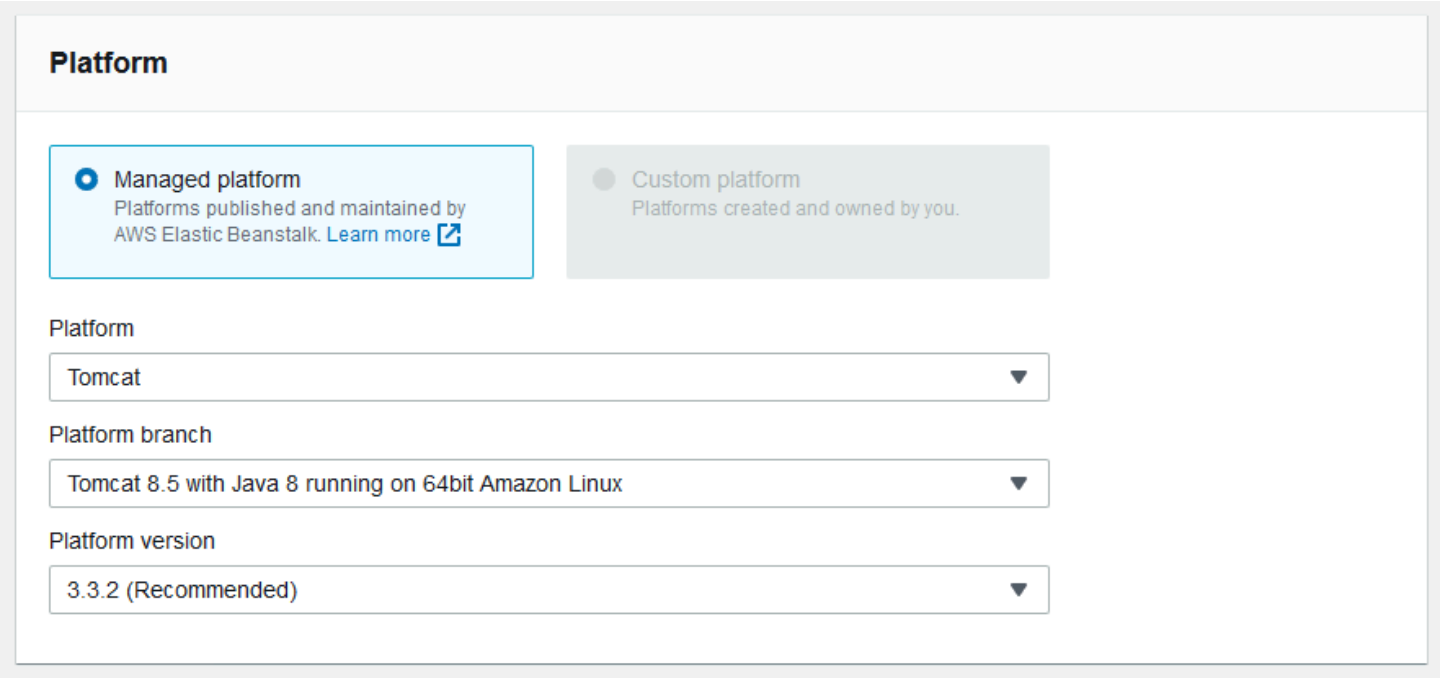
为新环境选择平台

您可以根据两种类型的平台创建新环境：

- 托管平台
- 自定义平台

托管平台

大多数情况下，您将为新环境使用 Elastic Beanstalk 托管平台。当新环境向导启动后，它会默认选择 Managed platform（托管式平台）选项。



选择平台、该平台内的平台分支以及分支中的特定平台版本。在选择平台分支时，默认情况下会选择分支中的推荐版本。此外，您可以选择之前使用过的任意平台版本。

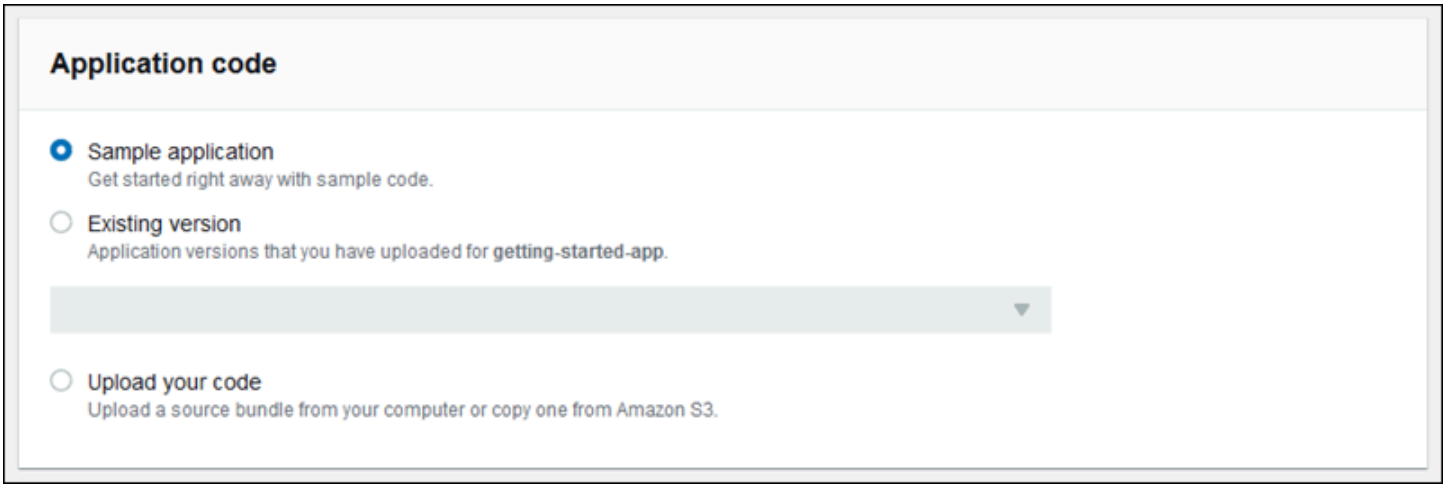
> **Note**
>
> 对于生产环境，我们建议您在受支持的平台分支中选择平台版本。有关平台分支状态的详细信息，请参阅 the section called “平台词汇表”中的平台分支 定义。

自定义平台

如果现成的平台满足不了需求，可以通过自定义平台创建新环境。要指定自定义平台，请选择 Custom platform（自定义平台）选项，然后选择其中一个可用的自定义平台。如果无自定义平台可用，则此选项将灰显。

提供应用程序代码

现在您已选择了要使用的平台，下一步是提供您的应用程序代码。

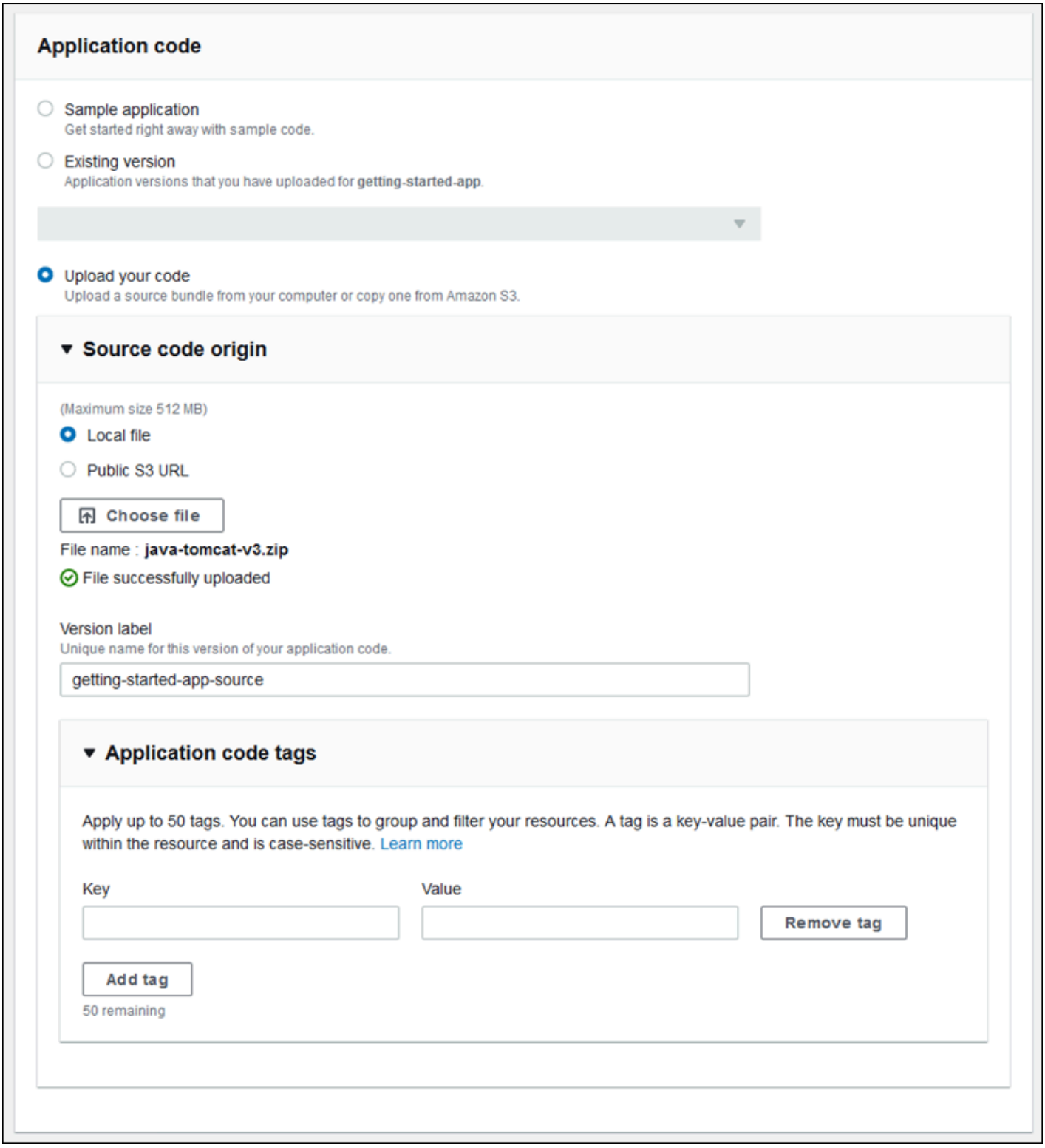


您有多种选择：

- 您可以使用 Elastic Beanstalk 为每个平台提供的示例应用程序。
- 您可以使用已部署到 Elastic Beanstalk 的代码。在 Application code（应用程序代码）部分中选择 Existing version（现有版本）和应用程序。
- 您可以上传新代码。选择上传代码，然后选择上传。您可以从本地文件上传新的应用程序代码，也可以指定包含应用程序代码的 Amazon S3 存储桶对应的 URL。

> **Note**
>
> 根据您选择的平台版本，您能够以 ZIP 源包、WAR 文件或纯文本 Docker 配置的形式上传应用程序。文件大小限制为 500 MB。

当您选择上传新代码时，还可以提供要与新代码关联的标签。有关标记应用程序版本的更多信息，请参阅the section called “标记应用程序版本”。



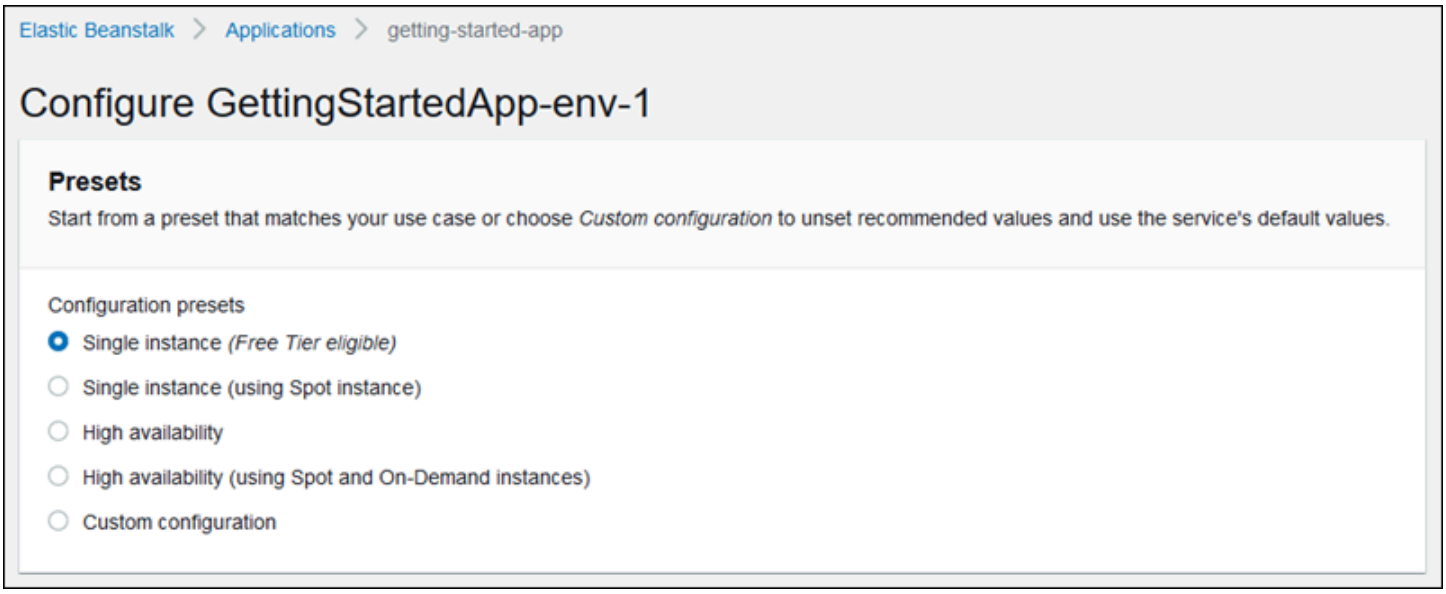
要使用默认配置选项快速创建环境，您现在可以选择 Create environment（创建环境）。选择 Configure more option（配置更多选项）以完成其他配置更改，如以下部分所述。

向导配置页面

当您选择 Configure more options（配置更多选项）时，向导会显示 Configure（配置）页面。在此页面上，您可以选择配置预设、更改您希望环境使用的平台版本或者为新环境选择特定的配置。

选择预设配置

在页面的 Presets（预设）部分中，Elastic Beanstalk 为不同的使用案例提供了多种配置预设。每个预设包含若干配置选项的建议值。



高可用性预设包括负载均衡器，建议用于生产环境。如果您需要可运行多个实例来实现高可用性并可根据负载进行调整的环境，请选择这些选项。建议将单实例预设主要用于开发。其中两种预设启用 Spot 实例请求。有关 Elastic Beanstalk 容量配置的详细信息，请参阅Auto Scaling 组。

最后一种预设是自定义配置，它删除了除角色设置之外的所有建议值，而是使用 API 默认值。如果您要部署带有用于设置配置选项的配置文件的源包，请选择此选项。如果修改 Low cost（低成本）或 High availability（高可用性）配置预置，则还将自动选择 Custom configuration（自定义配置）。

自定义您的配置

除了（而不是替代）选择配置预设之外，您还可以优化您的环境中的配置选项。Configure（配置）向导显示了多个配置类别。每个配置类别显示一组配置设置的值摘要。选择 Edit（编辑）以编辑此设置组。

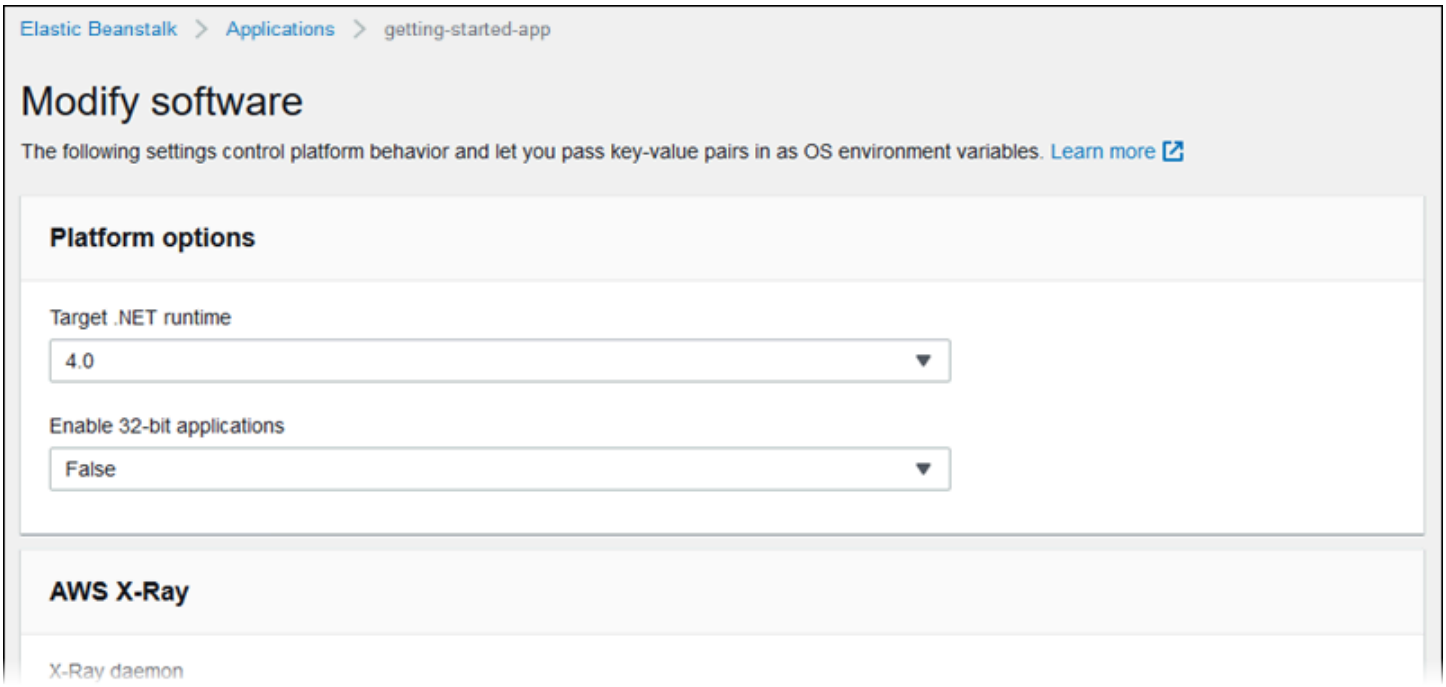
配置类别

- 软件设置

- 实例
- 容量
- 负载均衡器
- 滚动更新和部署
- 安全性
- 监控
- 托管更新
- 通知
- Network
- 数据库
- 标签
- 工作线程环境

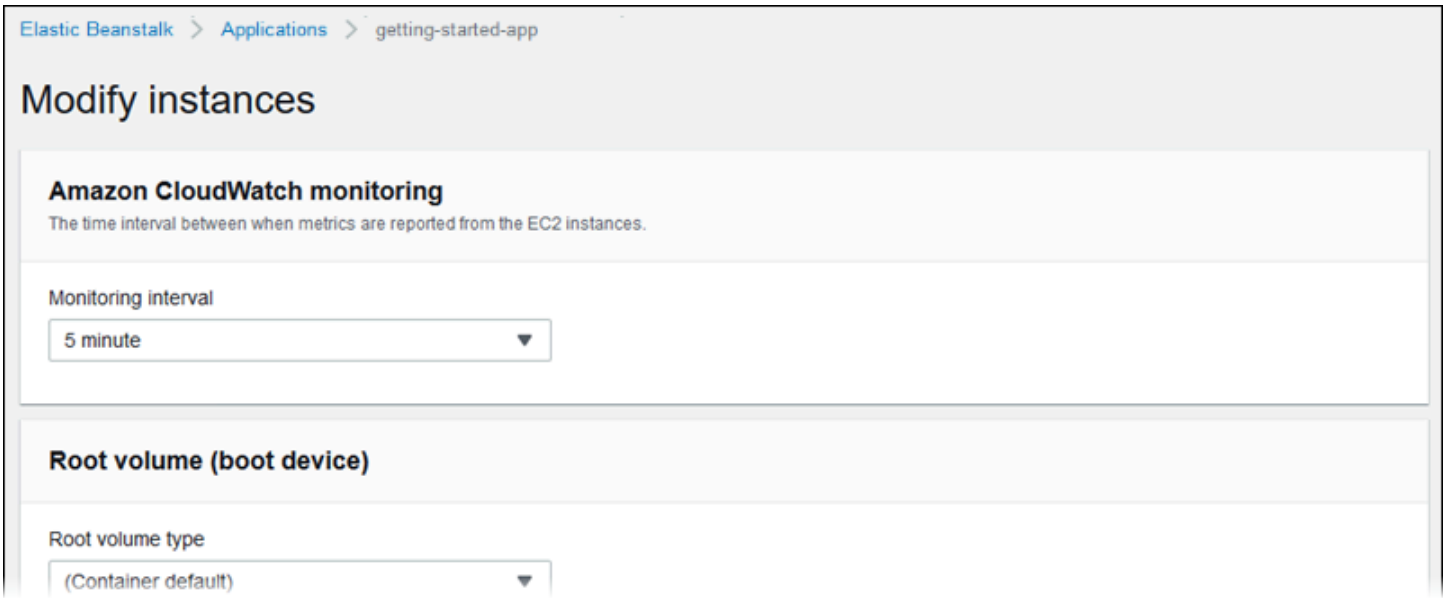
软件设置

使用 Modify software（修改软件）配置页面，在运行应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）实例上配置软件。您可以配置环境属性、AWS X-Ray 调试、实例日志的存储和流式传输以及特定于平台的设置。有关详细信息，请参阅the section called “环境属性和软件设置”。



实例

使用 Modify instances（修改实例）配置页面配置运行应用程序的 Amazon EC2 实例。有关详细信息，请参阅the section called “Amazon EC2 实例”。



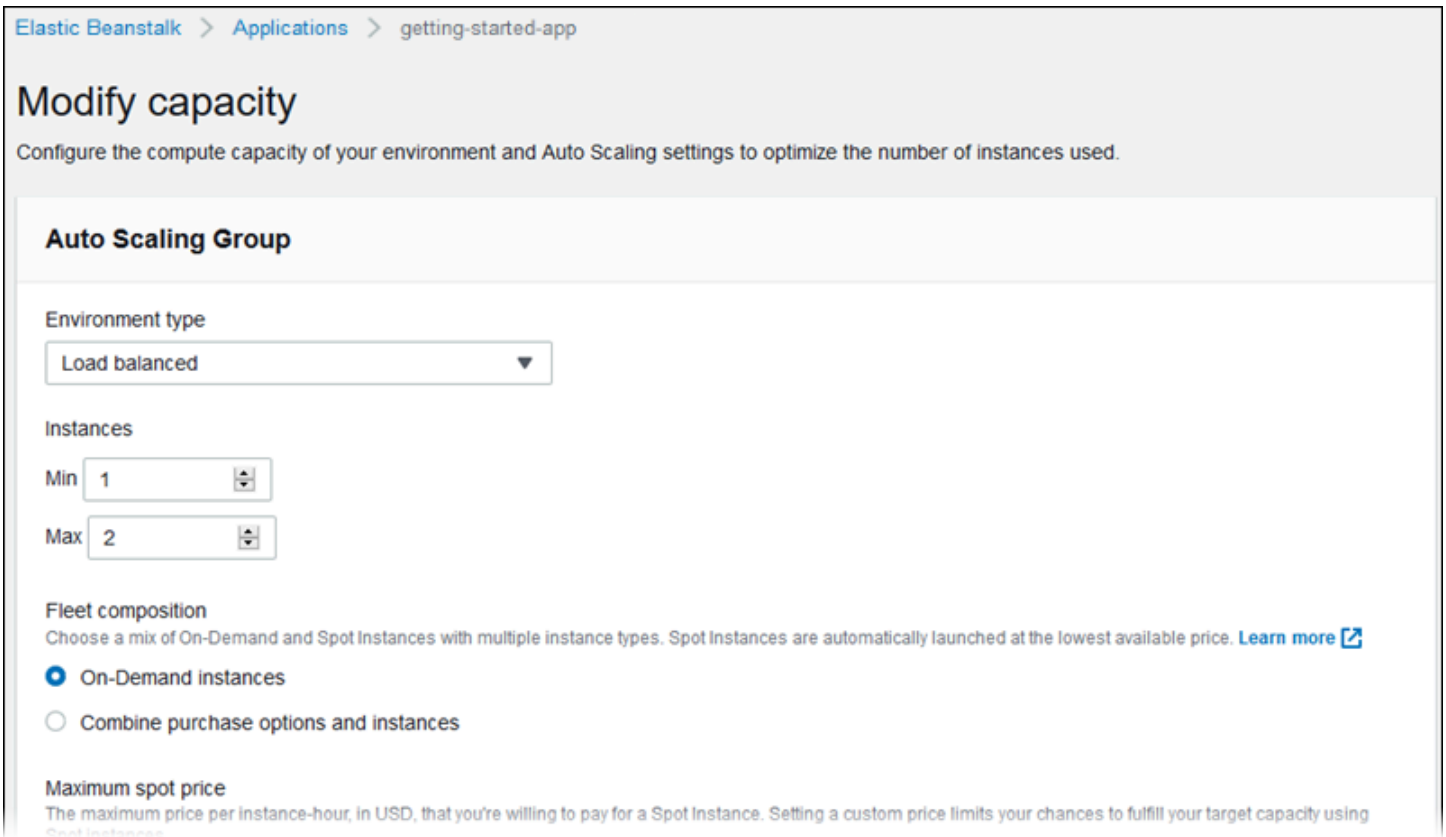
容量

使用 Modify capacity（修改容量）配置页面配置环境的计算容量和 Auto Scaling group（Auto Scaling 组）设置，以优化正在使用的实例的数量和类型。还可以根据触发器或计划更改环境容量。

负载均衡环境可运行多个实例来实现高可用性并防止在配置更新和部署期间停机。在负载均衡环境中，域名将映射到负载均衡器。在单个实例环境中，域名将映射到实例上的弹性 IP 地址。

> **Warning**
>
> 单实例环境不适用于生产。如果实例在部署期间变得不稳定，或者 Elastic Beanstalk 在配置更新期间终止并重新启动实例，则您的应用程序可能会在一段时间内不可用。可将单实例环境用于开发、测试或暂存。使用负载均衡环境进行生产。

有关环境容量设置的更多信息，请参阅 the section called “Auto Scaling 组” 和 the section called “Amazon EC2 实例”。

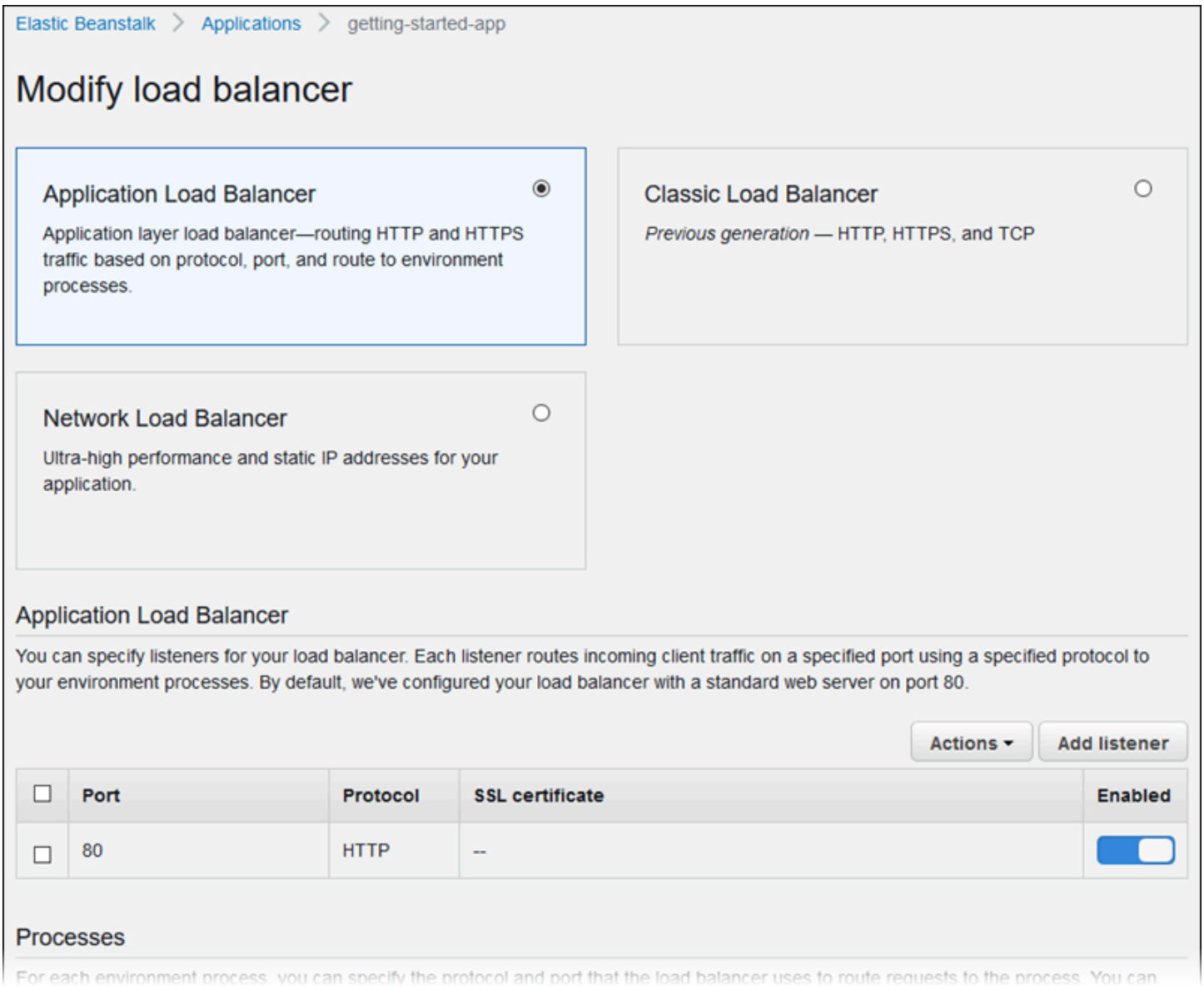


负载均衡器

使用 Modify load balancer（修改负载均衡器）配置页面选择负载均衡器类型和配置负载均衡器设置。在负载均衡环境中，环境的负载均衡器是发送到应用程序的所有流量的入口点。Elastic Beanstalk 支持多种类型的负载均衡器。默认情况下，Elastic Beanstalk 控制台会创建 Application Load Balancer 并将其配置为处理端口 80 上的 HTTP 流量。

> **Note**
>
> 您仅可以在环境创建期间选择环境的负载均衡器类型。

有关负载均衡器类型和设置的更多信息，请参阅the section called “负载均衡器”和the section called “HTTPS”。

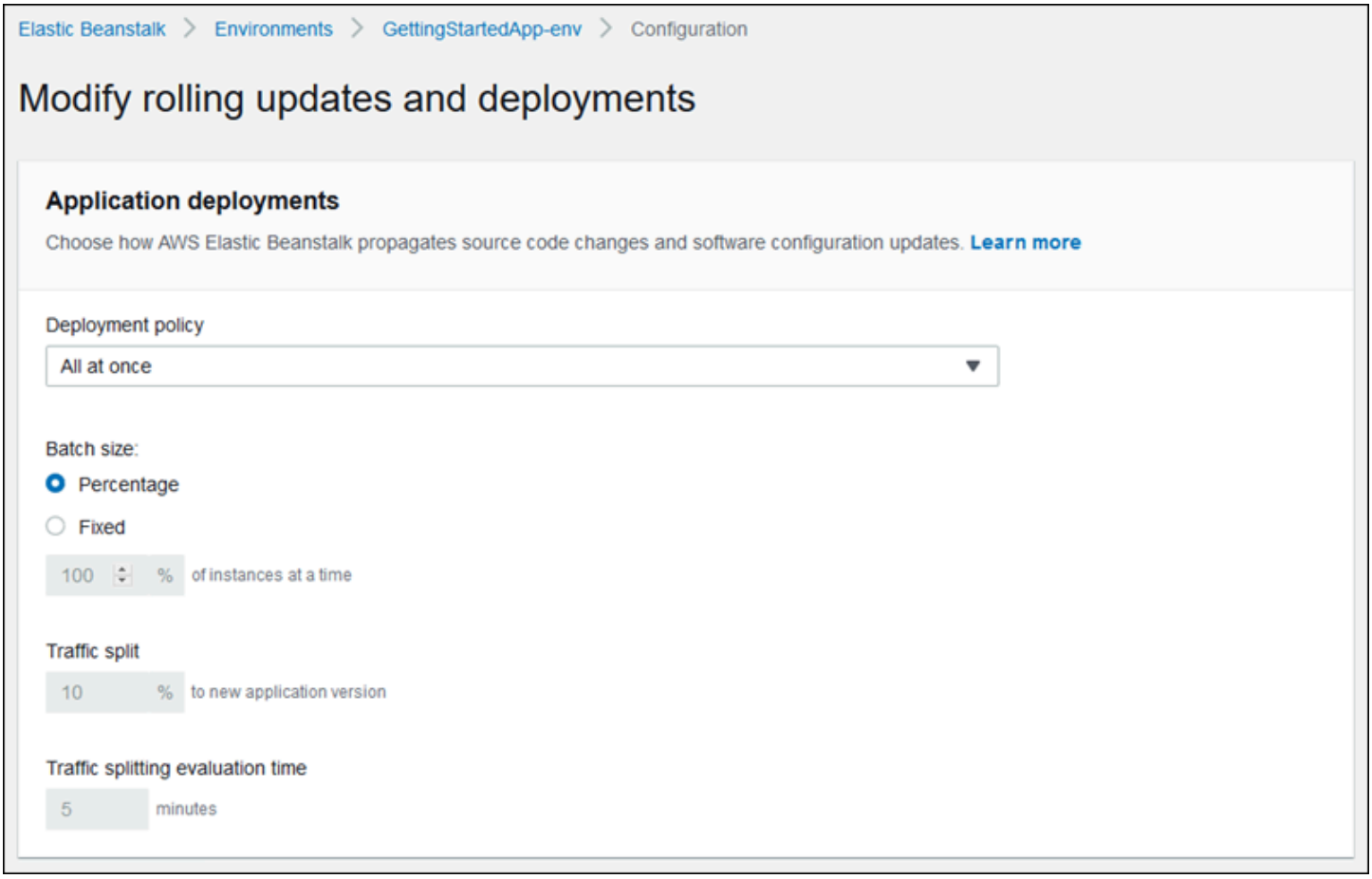


> **Note**
>
> 在创建环境控制台向导中禁用了经典负载均衡器（CLB）选项。如果某个现有的环境已经配置了经典负载均衡器，则可以使用 Elastic Beanstalk 控制台或 EB CLI 克隆现有环境，从而创建新环境。您还可以使用 EB CLI 或 AWS CLI 创建配置了经典负载均衡器的新环境。这些命令行工具将使用 CLB 创建一个新环境，即使您的账户中尚不存在该环境。

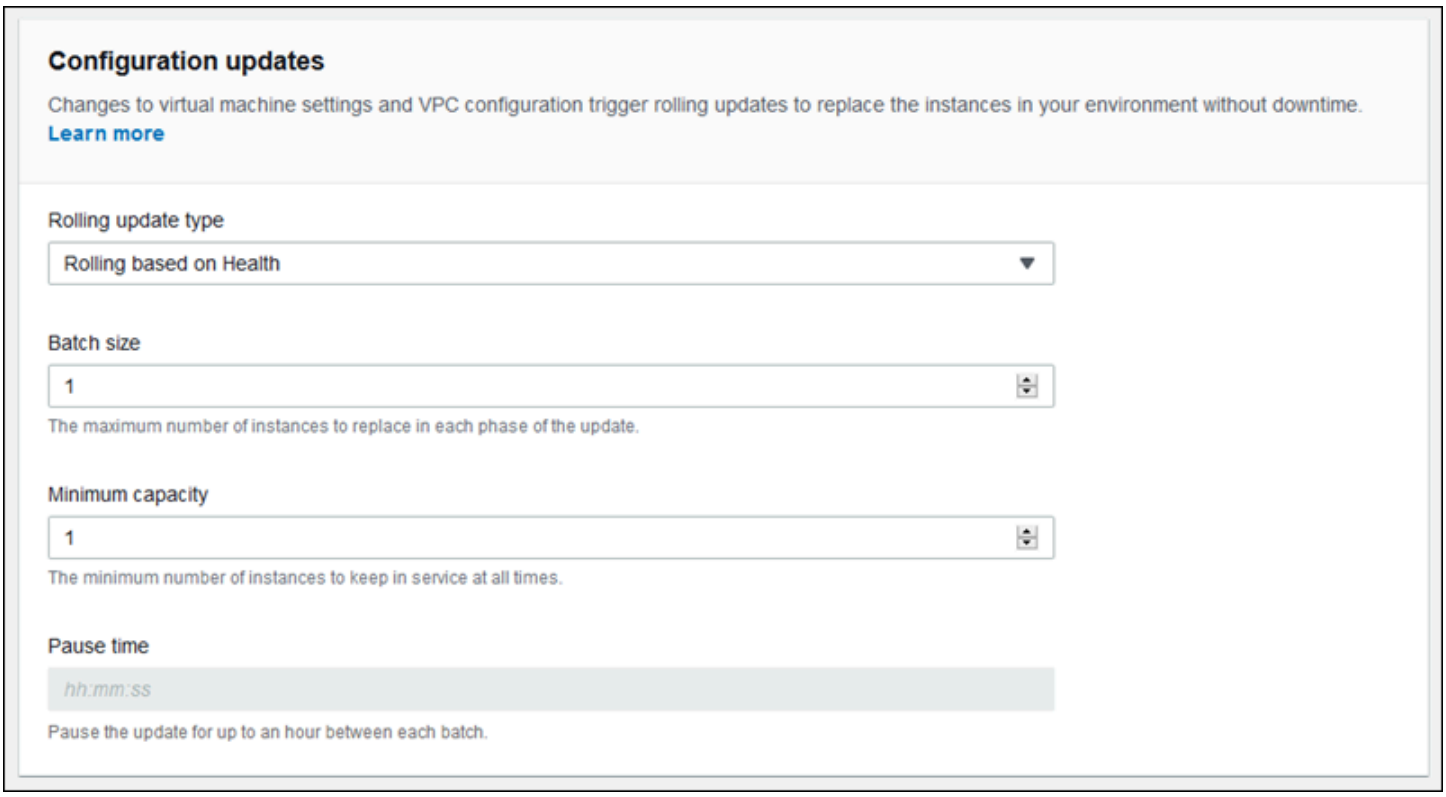
滚动更新和部署

使用 Modify rolling updates and deployments（修改滚动更新和部署）配置页面，配置 Elastic Beanstalk 如何处理环境的应用程序部署和配置更新。

当您上传更新后的应用程序源包并将其部署到您的环境时，会执行应用程序部署。有关配置部署的更多信息，请参阅the section called “部署选项”。



修改启动配置或 VPC 设置的配置更改需要终止您的环境中的所有实例并替换它们。有关设置更新类型和其他选项的更多信息，请参阅the section called “配置更改”。



安全性

使用 Configure service access（配置服务访问）页面配置服务和实例安全性设置。

有关 Elastic Beanstalk 安全概念的说明，请参阅权限。

首次在 Elastic Beanstalk 控制台中创建环境时，您必须创建一个具有默认权限集的 EC2 实例配置文件。如果 EC2 实例配置文件下拉列表未显示任何可供选择的值，请按下面的过程操作。此过程提供了创建角色的步骤，之后您可以为 EC2 实例配置文件选择该角色。

为 EC2 实例配置文件创建 IAM 角色

创建用于 EC2 实例配置文件选择的 IAM 角色

1. 选择查看权限详细信息。这将在 EC2 实例配置文件下拉列表下显示。

   这时会显示一个名为查看实例配置文件权限的模态窗口。此窗口将列出您需要附加到所创建的新 EC2 实例配置文件的托管式配置文件。此外还提供了一个用于启动 IAM 控制台的链接。
2. 选择窗口顶部显示的 IAM 控制台链接。
3. 请在 IAM 控制台的导航窗格中，选择 Roles（角色）。
4. 选择 Create role（创建角色）。

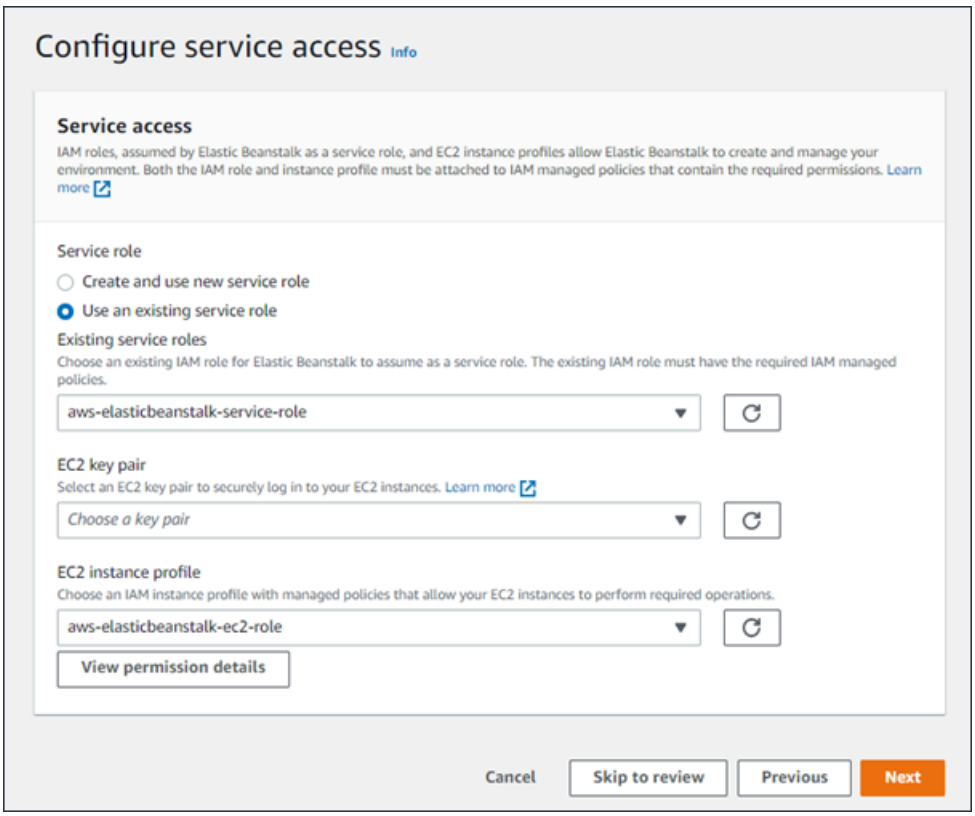
5. 在可信实体类型下，选择 AWS 服务。
6. 在 Use case（使用案例）下，选择 EC2。
7. 选择 Next（下一步）。
8. 附加适当的托管式策略。滚动查看实例配置文件权限模式窗口，以查看托管式策略。这些策略还将在此处列出：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`
9. 选择 Next（下一步）。
10. 输入角色的名称。
11. （可选）将标签添加到角色。
12. 选择 Create role（创建角色）。
13. 返回已打开的 Elastic Beanstalk 控制台窗口。
14. 关闭查看实例配置文件权限模态窗口。

> **Important**
>
> 不要关闭显示 Elastic Beanstalk 控制台的浏览器页面。

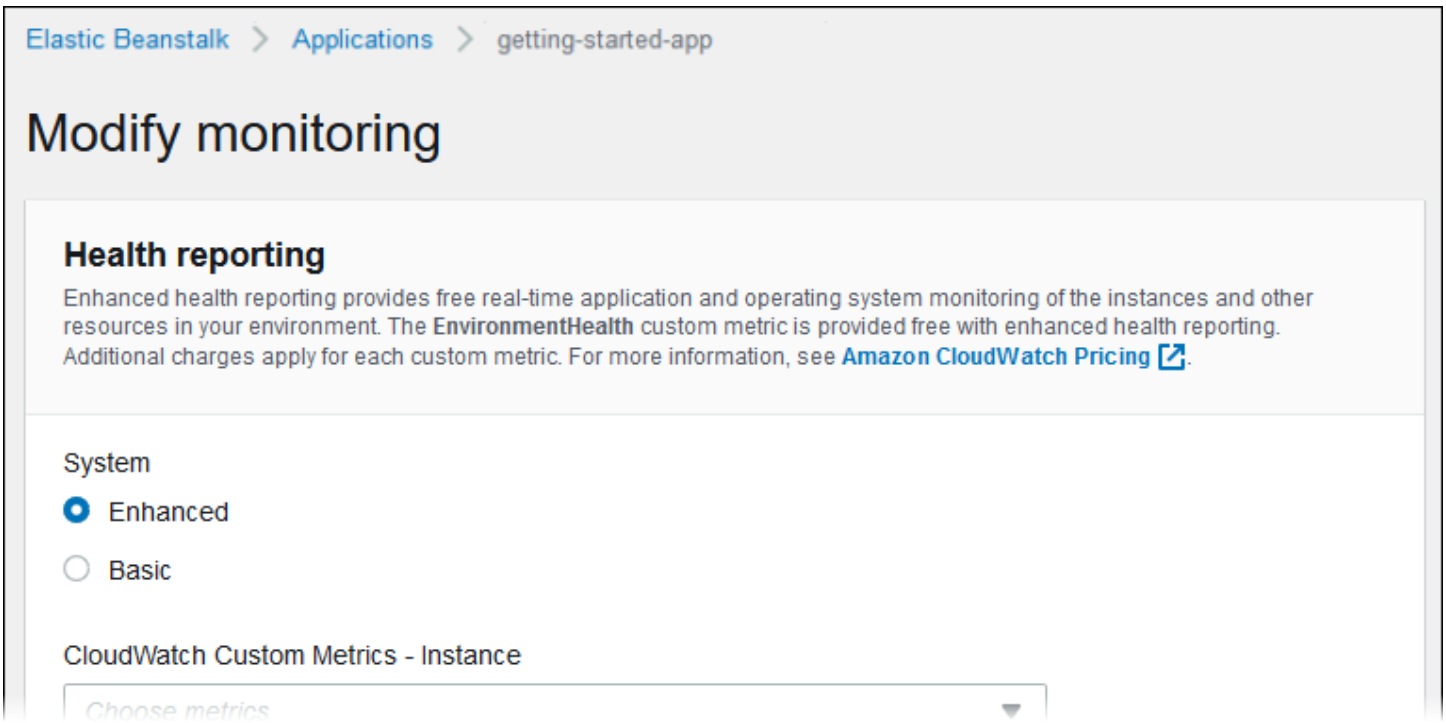
15. 选择 EC2 实例配置文件下拉列表旁边的  （刷新）。

    这将刷新下拉列表，以确保您刚刚创建的角色会在下拉列表中显示。



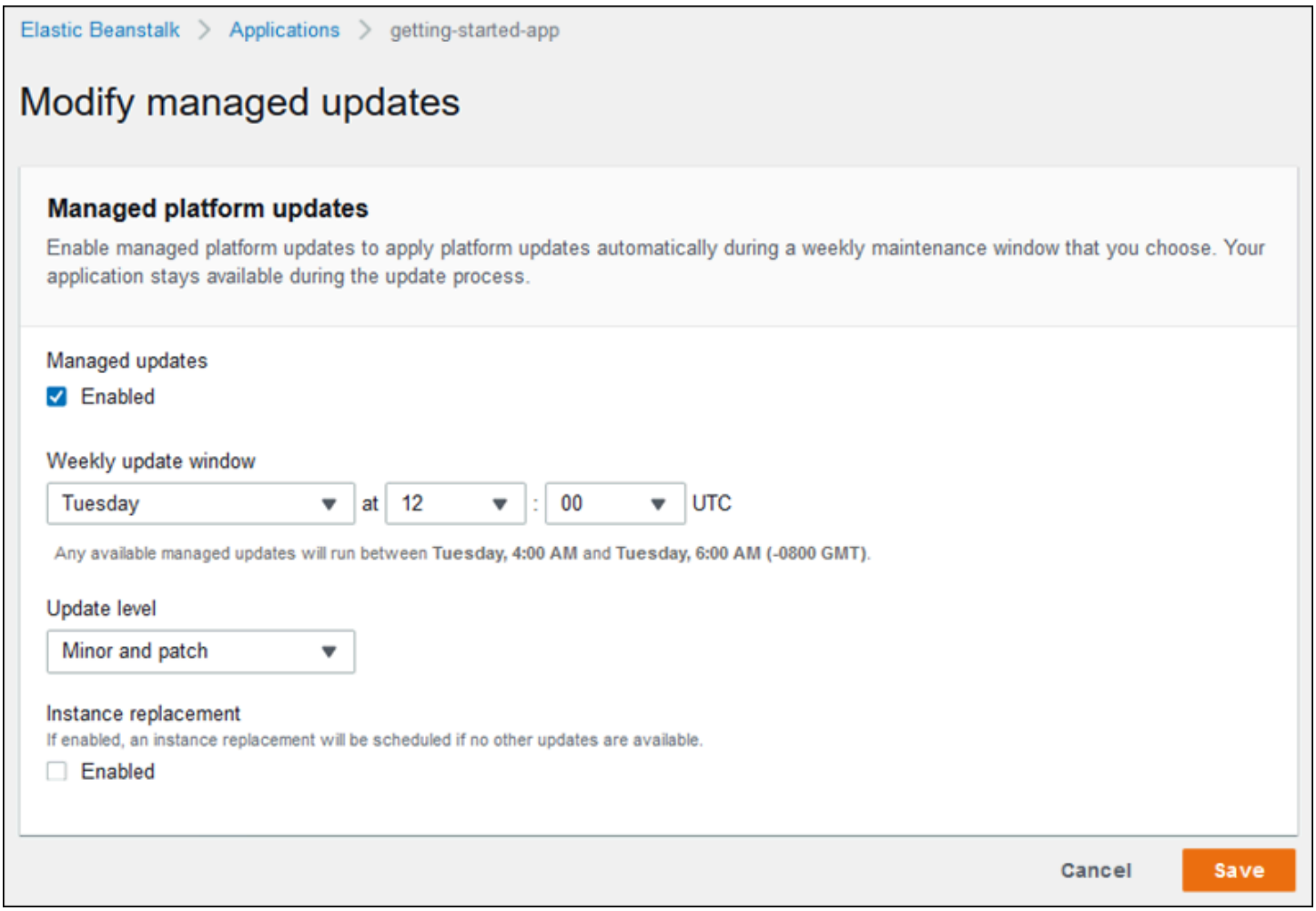
监控

使用 Modify monitoring（修改监控）配置页面可配置运行状况报告、监控规则和运行状况事件流。有关详细信息，请参阅the section called “启用增强型运行状况”、the section called “增强型运行状况规则”和the section called “流式传输环境运行状况”。



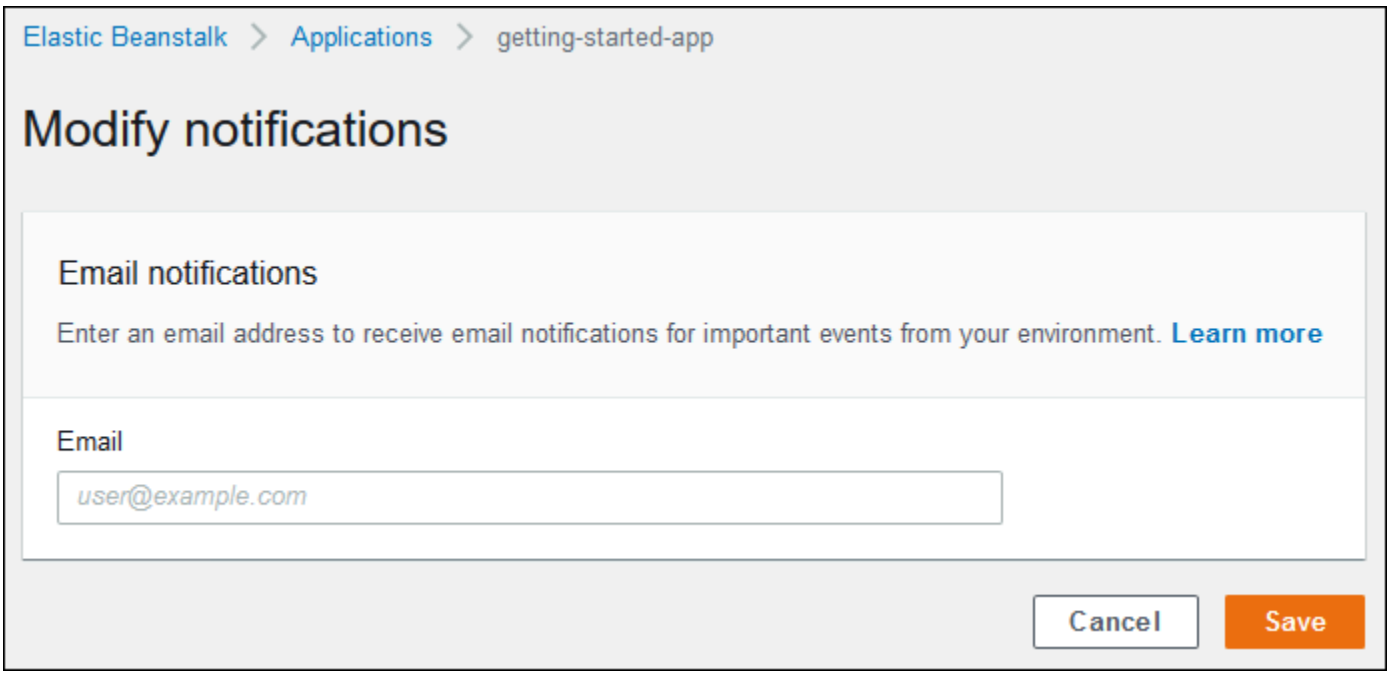
托管更新

使用 Modify managed updates（修改托管更新）配置页面可配置托管平台更新。您可以决定是否要启用它们、设置计划和配置其他属性。有关详细信息，请参阅the section called “托管更新”。



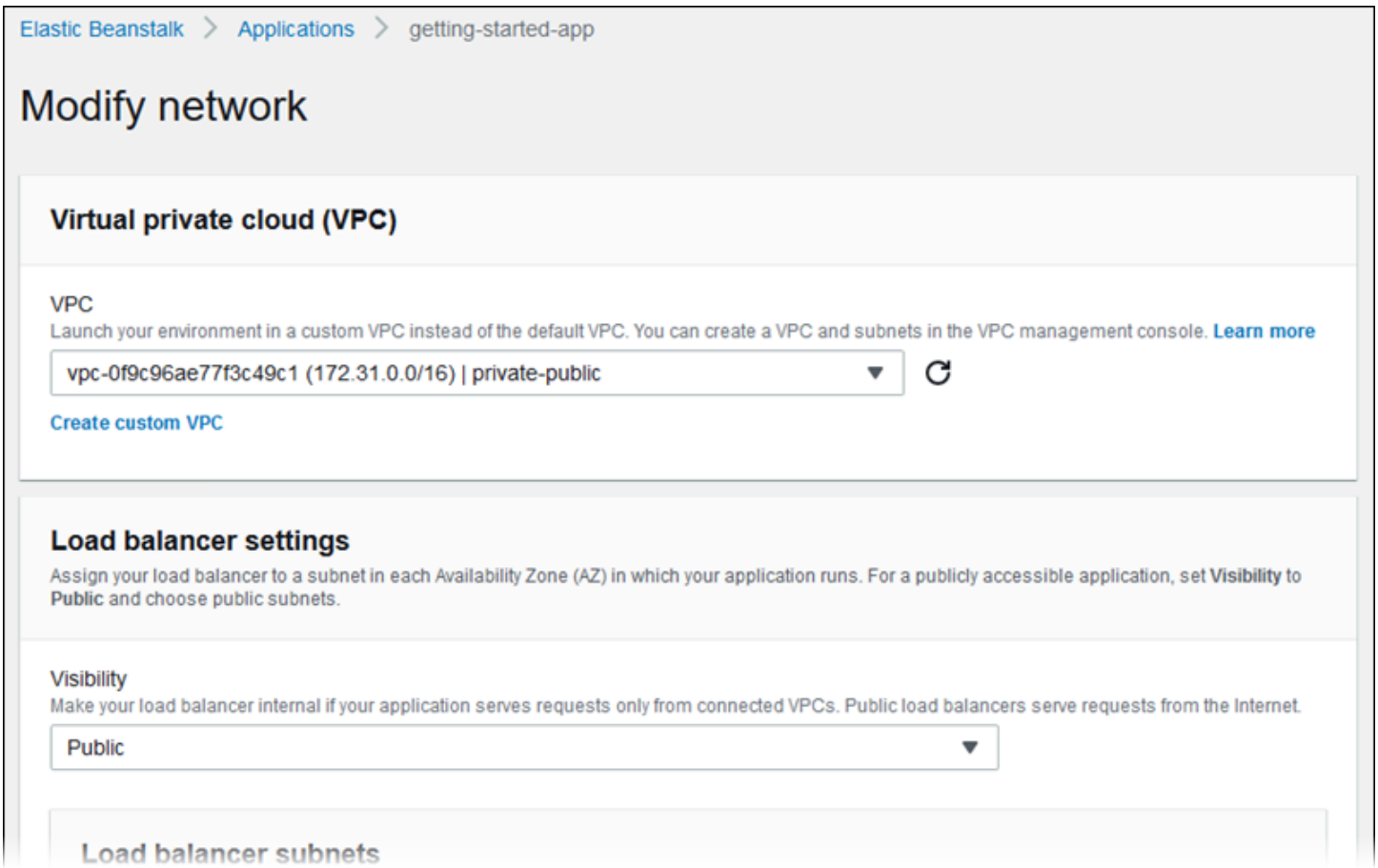
通知

使用 Modify notifications（修改通知）配置页面可指定用于接收来自环境的重要事件的电子邮件通知的电子邮件地址。



Network

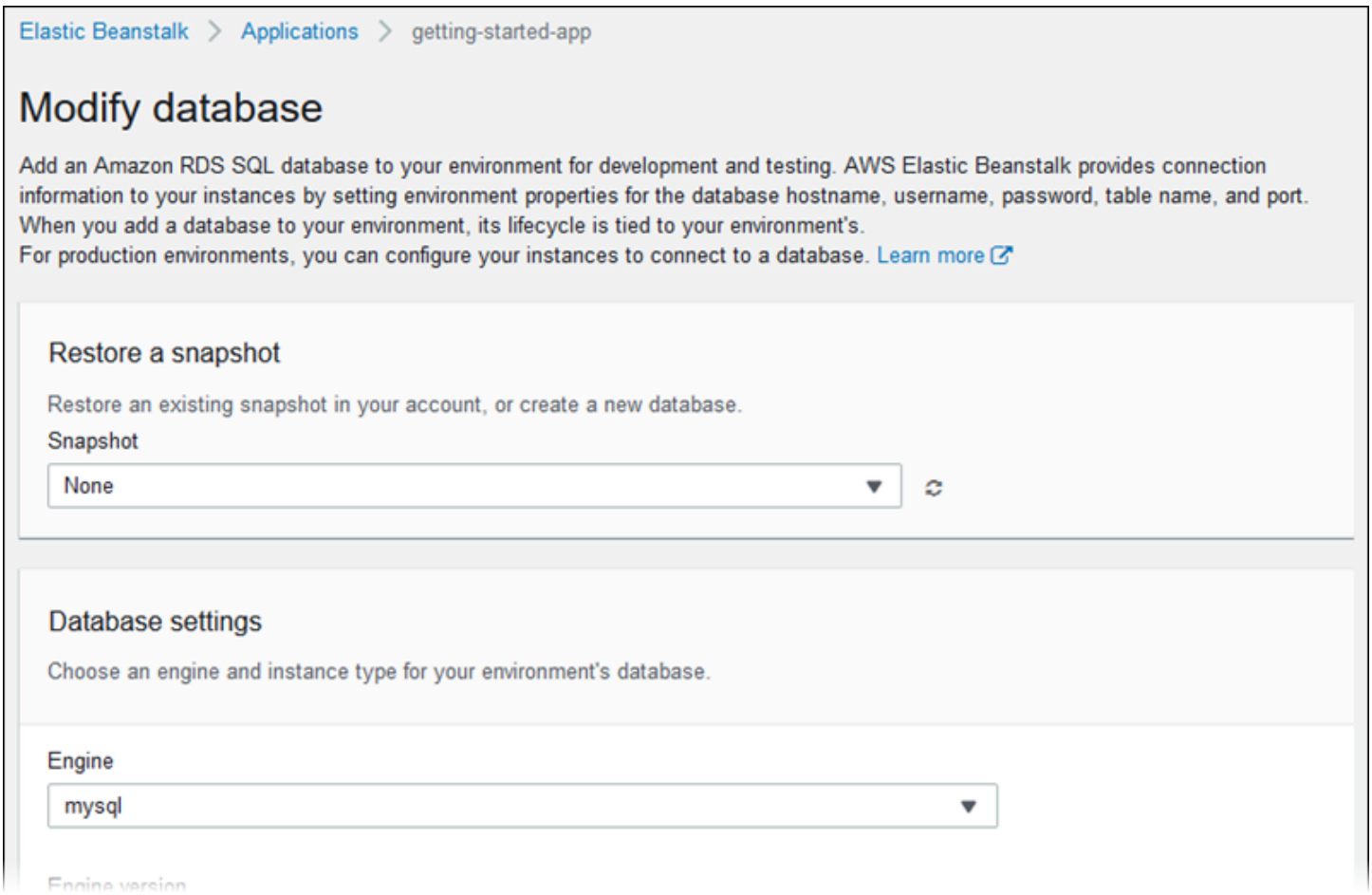
如果您已创建自定义 VPC，则可使用 Modify network（修改网络）配置页面将环境配置为使用该 VPC。如果您不选择 VPC，Elastic Beanstalk 将使用默认的 VPC 和子网。



数据库

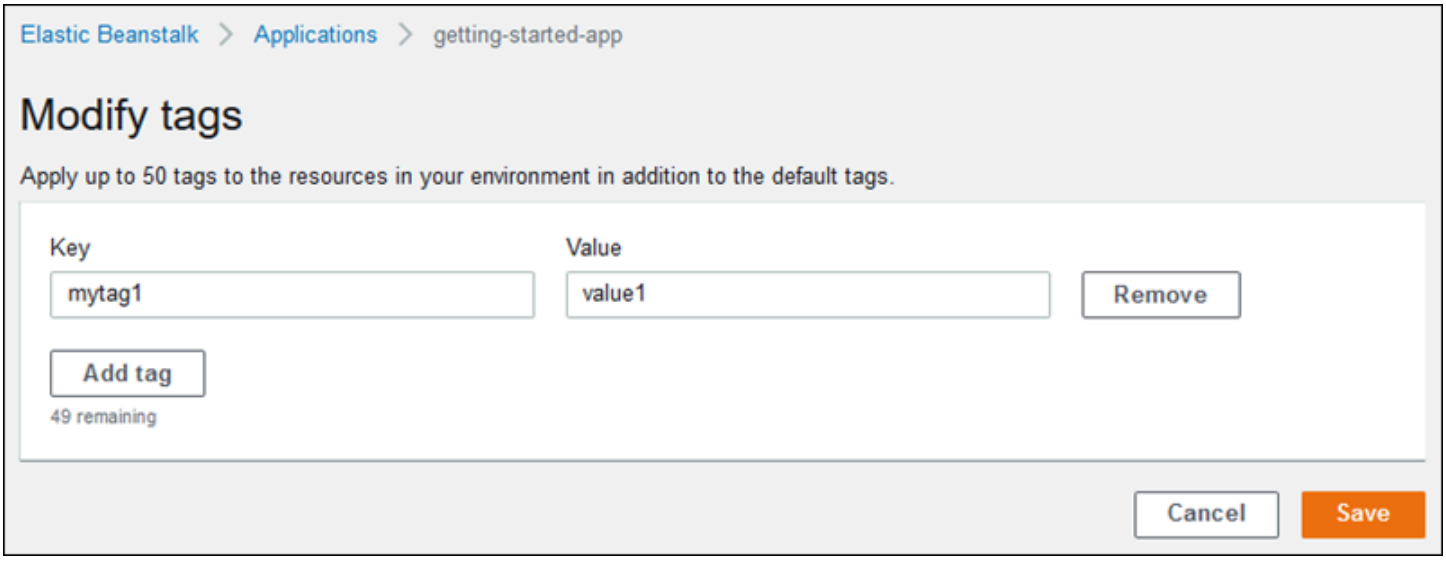
使用 Modify database（修改数据库）配置页面，将 Amazon Relational Database Service（Amazon RDS）数据库添加到环境中以进行开发和测试。Elastic Beanstalk 通过设置数据库主机名、用户名、密码、表名和端口的环境属性，为实例提供连接信息。

有关详细信息，请参阅the section called “数据库”。



标签

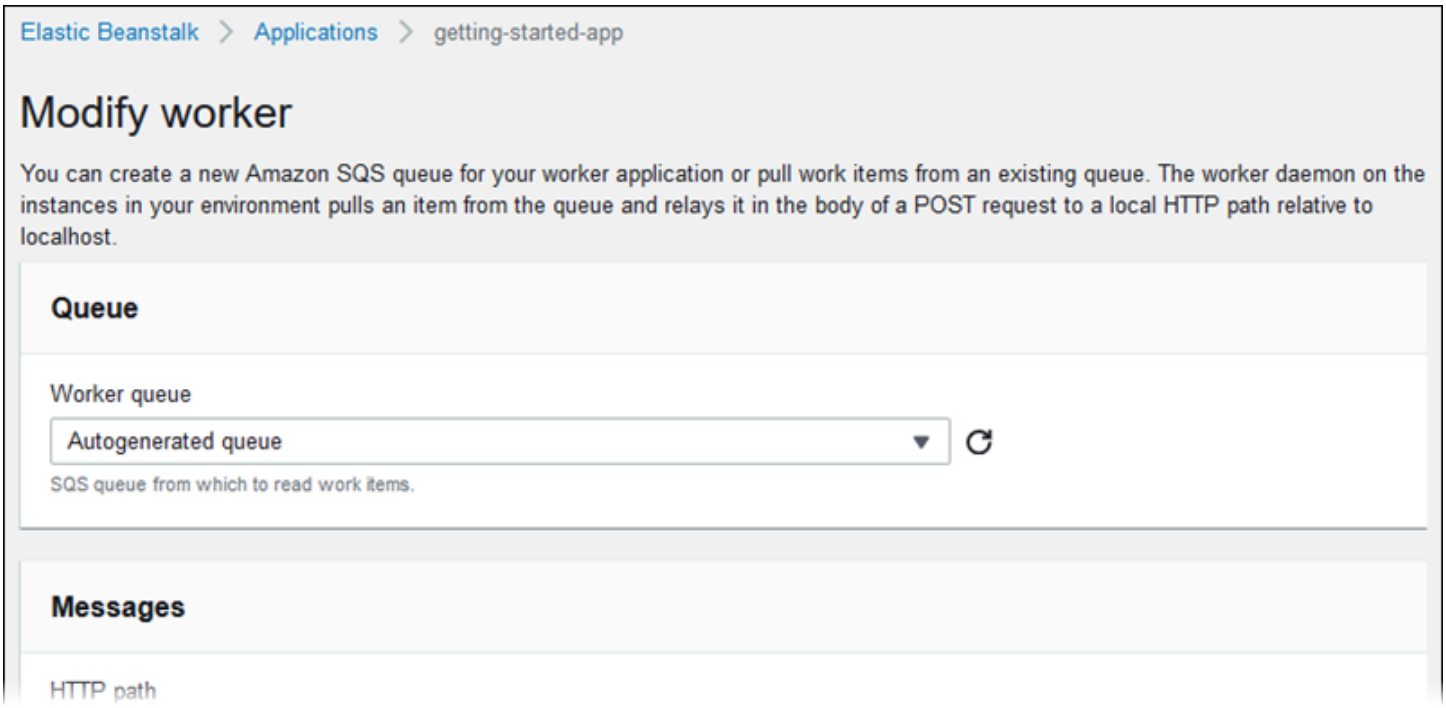
使用 Modify tags（修改标签）配置页面可将标签添加到环境中的资源。有关环境标记的更多信息，请参阅在 Elastic Beanstalk 环境中标记资源。



工作线程环境

如果要创建工作线程层环境，请使用 Modify worker（修改工作线程）配置页面来配置工作线程环境。环境中实例上的工作线程守护程序从 Amazon Simple Queue Service (Amazon SQS) 队列中提取项目，然后将它们作为发布消息中继到您的工作线程应用程序。您可以选择工作线程守护程序从中读取的 Amazon SQS 队列（自动生成的或现有的）。您还可以配置工作线程守护程序发送到您的应用程序的消息。

有关更多信息，请参阅the section called “工作线程环境”。



克隆 Elastic Beanstalk 环境

通过克隆现有的 Elastic Beanstalk 环境，您可以使用现有环境作为新环境的基础。例如，您可能希望创建一个克隆，以便使用原始环境平台所使用的较新版本的平台分支。Elastic Beanstalk 使用原始环境使用的环境设置来配置克隆。通过克隆现有环境而不是创建新环境，您无需手动配置使用 Elastic Beanstalk 服务所做的选项设置、环境变量和其他设置。Elastic Beanstalk 还会创建与原始环境关联的任何 AWS 资源的副本。

请务必注意以下情况：

- 在克隆过程中，Elastic Beanstalk 不会将数据从 Amazon RDS 复制到克隆。
- Elastic Beanstalk 不会将对资源的任何非托管更改包含在克隆中。您使用 Elastic Beanstalk 控制台、命令行工具或 API 以外的工具对 AWS 资源进行的更改被视为非托管更改。

- 用于入口的安全组被视为非托管更改。克隆的 Elastic Beanstalk 环境不会延续安全组进行入口，从而使环境对所有互联网流量开放。您需要为克隆的环境重新建立入口安全组。

您只能将环境克隆到相同平台分支的不同平台版本。不同的平台分支不能保证兼容。要使用不同的平台分支，您必须手动创建新的环境，部署应用程序代码，并对代码和选项进行必要的更改，以确保应用程序可在新平台分支上正常工作。

AWS 管理控制台

> **Important**
>
> 克隆的 Elastic Beanstalk 环境不会延续安全组进行入口，从而使环境对所有互联网流量开放。您需要为克隆的环境重新建立入口安全组。
> 您可以通过检查环境配置的偏差状态来查看可能无法克隆的资源。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的检测整个 CloudFormation 堆栈上的偏差。

克隆环境

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择操作。
4. 选择克隆环境。
5. 在 Clone environment（克隆环境）页面上，查看 Original Environment（原始环境）部分中的信息以验证您是否选择了要从其中创建克隆的环境。
6. 在 New Environment（新环境）部分中，您可以选择性地更改 Elastic Beanstalk 根据原始环境自动设置的 Environment name（环境名称）、Environment URL（环境 URL）、Description（描述）、Platform version（平台版本）和 Service role（服务角色）值。

> **Note**
>
> 如果原始环境中使用的平台版本与平台分支中推荐使用的版本不同，则会向您发送警告，建议您使用其他平台版本。选择平台版本，您可以在列表中看到推荐的平台版本，例如 3.3.2（推荐）。

7. 如果准备就绪，请选择 Clone（克隆）。

Elastic Beanstalk 命令行界面 (EB CLI)

> **Important**
>
> 克隆的 Elastic Beanstalk 环境不会延续安全组进行入口，从而使环境对所有互联网流量开放。您需要为克隆的环境重新建立入口安全组。
> 您可以通过检查环境配置的偏差状态来查看可能无法克隆的资源。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的检测整个 CloudFormation 堆栈上的偏差。

使用 eb clone 命令克隆正在运行的环境，如下所示。

```
~/workspace/my-app$ eb clone my-env1
Enter name for Environment Clone
(default is my-env1-clone): my-env2
Enter DNS CNAME prefix
(default is my-env1-clone): my-env2
```

您可以在克隆命令中指定源环境的名称，或留空以克隆当前项目文件夹的默认环境。EB CLI 会提示您为新环境输入名称和 DNS 前缀。

默认情况下，eb clone 会使用源环境平台的最新可用版本创建新环境。要强制 EB CLI 使用相同的版本 (即使有新版本可用时)，请使用 `--exact` 选项。

```
~/workspace/my-app$ eb clone --exact
```

有关此命令的更多信息，请参阅 eb clone。

终止 Elastic Beanstalk 环境

您可以使用 Elastic Beanstalk 控制台终止正在运行的 AWS Elastic Beanstalk 环境。通过这样做，您可以避免因未使用的 AWS 资源产生费用。

> **Note**
>
> 稍后，您始终都可以使用相同的版本启动新的环境。

如果环境中有要保留的数据，请在终止环境之前将数据库删除策略设置为 `Retain`。这使数据库能够在 Elastic Beanstalk 之外运行。之后，任何 Elastic Beanstalk 环境都必须作为外部数据库连接到它。如果要在不保持数据库运行的情况下备份数据，请将删除策略设置为在终止环境之前拍摄数据库快照。有关更多信息，请参阅本指南的配置环境章节中的数据库生命周期。

Elastic Beanstalk 可能无法终止您的环境。一个常见原因是，另一个环境的安全组在您要终止的环境的安全组上有依赖项。有关如何避免此问题的说明，请参阅本指南的 EC2 实例页面中的安全组。

> **Important**
>
> 如果您终止环境，则还必须删除您创建的任何 CNAME 映射，因为其他客户可能会重用可用的主机名。请务必删除指向已终止环境的 DNS 记录，以防出现 悬空 DNS 条目。悬空 DNS 条目可能会使指向您的域的互联网流量出现安全漏洞，此外还可能带来其他风险。
> 有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的 防止 Route 53 中悬挂委派记录。您还可以访问《AWS 安全博客》中的 针对 Amazon CloudFront 请求的增强域保护，了解有关悬空 DNS 条目的更多信息。

Elastic Beanstalk 控制台

终止环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

> **Note**
>
> 终止环境时，与已终止环境相关联的别名记录可供任何人使用。

Elastic Beanstalk 需要几分钟时间才能终止环境中运行的 AWS 资源。

AWS CLI

终止环境

- 运行以下命令。

```
$ aws elasticbeanstalk terminate-environment --environment-name my-env
```

API

终止环境

- 按照以下参数请求 `TerminateEnvironment`：

  `EnvironmentName = SampleAppEnv`

```
https://elasticbeanstalk.us-west-2.amazon.com/?EnvironmentName=SampleAppEnv
&Operation=TerminateEnvironment
&AuthParams
```

使用 AWS CLI 创建 Elastic Beanstalk 环境

有关 Elastic Beanstalk AWS CLI 命令的详细信息，请参阅命令参考。AWS CLI

1. 检查是否为环境提供了别名记录。

```
$ aws elasticbeanstalk check-dns-availability --cname-prefix my-cname
```

```
{
    "Available": true,
    "FullyQualifiedCNAME": "my-cname.elasticbeanstalk.com"
}
```

2. 请确保您的应用程序版本存在。

```
$ aws elasticbeanstalk describe-application-versions --application-name my-app --
version-label v1
```

如果您的源还没有应用程序版本，请创建。例如，以下命令将从 Amazon Simple Storage Service (Amazon S3) 中的源包创建一个应用程序版本。

```
$ aws elasticbeanstalk create-application-version --application-name my-app --
version-label v1 --source-bundle S3Bucket=DOC-EXAMPLE-BUCKET,S3Key=my-source-
bundle.zip
```

3. 为应用程序创建配置模板。

```
$ aws elasticbeanstalk create-configuration-template --application-name my-app --
template-name v1 --solution-stack-name "64bit Amazon Linux 2015.03 v2.0.0 running
 Ruby 2.2 (Passenger Standalone)"
```

4. 创建环境。

```
$ aws elasticbeanstalk create-environment --cname-prefix my-cname --application-
name my-app --template-name v1 --version-label v1 --environment-name v1clone --
option-settings file://options.txt
```

选项设置在 options.txt 文件中定义：

```
[
    {
        "Namespace": "aws:autoscaling:launchconfiguration",
        "OptionName": "IamInstanceProfile",
        "Value": "aws-elasticbeanstalk-ec2-role"
    }
]
```

上面的选项设置定义了 IAM 实例配置文件。您可以指定 ARN 或配置文件名称。

5. 确定新环境状态是否为“Green and Ready”。

```
$ aws elasticbeanstalk describe-environments --environment-names my-env
```

如果新环境的状态不是“Green and Ready”，您应决定是重试此操作，还是使环境保持当前状态，以执行调查。完成后，请确保终止环境，并清除所有未使用的资源。

> **Note**
>
> 如果环境没有在合理的时间内启动，您可调整超时周期。

使用 API 创建 Elastic Beanstalk 环境

1. 按照以下参数请求 `CheckDNSAvailability`：

   - `CNAMEPrefix` = `SampleApp`

   Example

   ```
   https://elasticbeanstalk.us-east-2.amazonaws.com/?CNAMEPrefix=sampleapplication
   &Operation=CheckDNSAvailability
   &AuthParams
   ```

2. 使用以下参数调用 `DescribeApplicationVersions`：

   - `ApplicationName` = `SampleApp`
   - `VersionLabel` = `Version2`

   Example

   ```
   https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
   &VersionLabel=Version2
   &Operation=DescribeApplicationVersions
   &AuthParams
   ```

3. 使用以下参数调用 `CreateConfigurationTemplate`：

   - `ApplicationName` = `SampleApp`
   - `TemplateName` = `MyConfigTemplate`

- SolutionStackName = 64bit%20Amazon%20Linux%202015.03%20v2.0.0%20running%20Ruby%202.2%20(Passenger%20Standalone)

Example

```
https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
&TemplateName=MyConfigTemplate
&Operation=CreateConfigurationTemplate
&SolutionStackName=64bit%20Amazon%20Linux%202015.03%20v2.0.0%20running%20Ruby
%202.2%20(Passenger%20Standalone)
&AuthParams
```

4. 使用以下参数集之一调用 CreateEnvironment：

   a. 为 Web 服务器环境层使用以下内容：

   - EnvironmentName = SampleAppEnv2
   - VersionLabel = Version2
   - Description = description
   - TemplateName = MyConfigTemplate
   - ApplicationName = SampleApp
   - CNAMEPrefix = sampleapplication
   - OptionSettings.member.1.Namespace = aws:autoscaling:launchconfiguration
   - OptionSettings.member.1.OptionName = IamInstanceProfile
   - OptionSettings.member.1.Value = aws-elasticbeanstalk-ec2-role

   Example

```
https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
&VersionLabel=Version2
&EnvironmentName=SampleAppEnv2
&TemplateName=MyConfigTemplate
&CNAMEPrefix=sampleapplication
&Description=description
&Operation=CreateEnvironment
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
```

```
&OptionSettings.member.1.OptionName=IamInstanceProfile
&OptionSettings.member.1.Value=aws-elasticbeanstalk-ec2-role
&AuthParams
```

b. 为工作程序环境层使用以下内容：

- EnvironmentName = SampleAppEnv2
- VersionLabel = Version2
- Description = description
- TemplateName = MyConfigTemplate
- ApplicationName = SampleApp
- Tier = Worker
- OptionSettings.member.1.Namespace = aws:autoscaling:launchconfiguration
- OptionSettings.member.1.OptionName = IamInstanceProfile
- OptionSettings.member.1.Value = aws-elasticbeanstalk-ec2-role
- OptionSettings.member.2.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.2.OptionName = WorkerQueueURL
- OptionSettings.member.2.Value = sqsd.elasticbeanstalk.us-east-2.amazonaws.com
- OptionSettings.member.3.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.3.OptionName = HttpPath
- OptionSettings.member.3.Value = /
- OptionSettings.member.4.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.4.OptionName = MimeType
- OptionSettings.member.4.Value = application/json
- OptionSettings.member.5.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.5.OptionName = HttpConnections
- OptionSettings.member.5.Value = 75
- OptionSettings.member.6.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.6.OptionName = ConnectTimeout
- OptionSettings.member.6.Value = 10
- OptionSettings.member.7.Namespace = aws:elasticbeanstalk:sqsd

- OptionSettings.member.7.OptionName = InactivityTimeout
- OptionSettings.member.7.Value = 10
- OptionSettings.member.8.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.8.OptionName = VisibilityTimeout
- OptionSettings.member.8.Value = 60
- OptionSettings.member.9.Namespace = aws:elasticbeanstalk:sqsd
- OptionSettings.member.9.OptionName = RetentionPeriod
- OptionSettings.member.9.Value = 345600

Example

```
https://elasticbeanstalk.us-east-2.amazonaws.com/?ApplicationName=SampleApp
&VersionLabel=Version2
&EnvironmentName=SampleAppEnv2
&TemplateName=MyConfigTemplate
&Description=description
&Tier=Worker
&Operation=CreateEnvironment
&OptionSettings.member.1.Namespace=aws%3Aautoscaling%3Alaunchconfiguration
&OptionSettings.member.1.OptionName=IamInstanceProfile
&OptionSettings.member.1.Value=aws-elasticbeanstalk-ec2-role
&OptionSettings.member.2.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.2.OptionName=WorkerQueueURL
&OptionSettings.member.2.Value=sqsd.elasticbeanstalk.us-east-2.amazonaws.com
&OptionSettings.member.3.Namespace=aws%3elasticbeanstalk%3sqsd
&OptionSettings.member.3.OptionName=HttpPath
&OptionSettings.member.3.Value=%2F
&OptionSettings.member.4.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.4.OptionName=MimeType
&OptionSettings.member.4.Value=application%2Fjson
&OptionSettings.member.5.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.5.OptionName=HttpConnections
&OptionSettings.member.5.Value=75
&OptionSettings.member.6.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.6.OptionName=ConnectTimeout
&OptionSettings.member.6.Value=10
&OptionSettings.member.7.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.7.OptionName=InactivityTimeout
&OptionSettings.member.7.Value=10
```

```
&OptionSettings.member.8.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.8.OptionName=VisibilityTimeout
&OptionSettings.member.8.Value=60
&OptionSettings.member.9.Namespace=aws%3Aelasticbeanstalk%3Asqsd
&OptionSettings.member.9.OptionName=RetentionPeriod
&OptionSettings.member.9.Value=345600
&AuthParams
```

构建 Launch Now URL

您可以构建自定义 URL，以便任何人都可以在 AWS Elastic Beanstalk中快速部署并运行预先确定的 Web 应用程序 此 URL 称为 Launch Now URL。例如，您可能需要一个 Launch Now URL 来演示构建为在 Elastic Beanstalk 上运行的 Web 应用程序。利用 Launch Now URL，您可以使用参数提前将所需信息添加到“创建应用程序”向导。您将该信息添加到向导后，任何人只需执行几个步骤，即可使用 URL 链接通过您的 Web 应用程序源启动 Elastic Beanstalk 环境。这意味着用户无需手动上载或指定应用程序源捆绑包的位置，也无需向该向导提供任何其他信息。

Launch Now URL 为 Elastic Beanstalk 提供创建应用程序所需的最少信息：应用程序名称、解决方案堆栈、实例类型和环境类型。Elastic Beanstalk 对未在自定义 Launch Now URL 中显式指定的其他配置详细信息使用默认值。

Launch Now URL 使用标准 URL 语法。有关更多信息，请参阅 RFC 3986 - 统一资源标识符 (URI)：一般语法。

URL 参数

URL 必须包含以下区分大小写的参数：

- 区域-指定 AWS 区域。有关 Elastic Beanstalk 支持的区域列表，请参阅 AWS 一般参考 中的 AWS Elastic Beanstalk 端点和配额。
- applicationName – 指定应用程序的名称。Elastic Beanstalk 会在 Elastic Beanstalk 控制台中显示应用程序名称以便与其他应用程序进行区分。默认情况下，应用程序名称还会形成环境名称和环境 URL 的基础。
- platform – 指定将用于环境的平台版本。使用以下方法之一，然后对您选择的项进行 URL 编码：
  - 指定不带版本的平台 ARN。Elastic Beanstalk 选择相应平台主要版本的最新平台版本。例如，要选择最新的 Python 3.6 平台版本，请指定 `Python 3.6 running on 64bit Amazon Linux`。
  - 指定平台名称。Elastic Beanstalk 选择平台最新语言（例如，`Python`）运行时的最新版本。

有关所有可用平台及其版本的描述，请参阅Elastic Beanstalk 支持的平台。

您可以使用 AWS Command Line Interface（AWS CLI）获取所有可用平台版本的列表及其相应的 ARN。`list-platform-versions` 命令列出有关所有可用平台版本的详细信息。使用 `--filters` 参数来缩小列表的范围。例如，您可以设置列表的范围来仅列出特定语言的平台版本。

以下示例查询所有 Python 平台版本，并将输出通过管道传送给一系列命令。结果是平台版本 ARN 的列表（没有 */version* 结尾），采用适合用户阅读的格式，没有 URL 编码。

```
$ aws elasticbeanstalk list-platform-versions --filters
 'Type="PlatformName",Operator="contains",Values="Python"' | grep PlatformArn | awk -
F '"' '{print $4}' | awk -F '/' '{print $2}'
Preconfigured Docker - Python 3.4 running on 64bit Debian
Preconfigured Docker - Python 3.4 running on 64bit Debian
Python 2.6 running on 32bit Amazon Linux
Python 2.6 running on 32bit Amazon Linux 2014.03
...
Python 3.6 running on 64bit Amazon Linux
```

以下示例将 Perl 命令添加到上一个示例，用于对输出进行 URL 编码。

```
$ aws elasticbeanstalk list-platform-versions --filters
 'Type="PlatformName",Operator="contains",Values="Python"' | grep PlatformArn | awk
 -F '"' '{print $4}' | awk -F '/' '{print $2}' | perl -MURI::Escape -ne 'chomp;print
 uri_escape($_),"\n"'
Preconfigured%20Docker%20-%20Python%203.4%20running%20on%2064bit%20Debian
Preconfigured%20Docker%20-%20Python%203.4%20running%20on%2064bit%20Debian
Python%202.6%20running%20on%2032bit%20Amazon%20Linux
Python%202.6%20running%20on%2032bit%20Amazon%20Linux%202014.03
...
Python%203.6%20running%20on%2064bit%20Amazon%20Linux
```

Launch Now URL 可以选择包含以下参数。如果您不在 Launch Now URL 中包含可选参数，Elastic Beanstalk 将使用默认值来创建和运行应用程序。如果不包含该sourceBundleUrl参数，Elastic Beanstalk 将使用指定平台的默认示例应用程序。

- sourceBundleUrl— 以 URL 格式指定 Web 应用程序源包的位置。例如，如果您将源包上传到 Amazon S3 存储桶，则可以将sourceBundleUrl参数的值指定为`https://mybucket.s3.amazonaws.com/myobject`。

> **Note**
>
> 您可以将sourceBundleUrl参数的值指定为 HTTP 网址，但用户的 Web 浏览器将通过应用 HTML URL 编码根据需要转换字符。

- environmentType – 指定环境是负载均衡的可扩展实例，还是仅为单个实例。有关更多信息，请参阅环境类型。您可以指定 `LoadBalancing` 或 `SingleInstance` 作为参数值。
- tierName – 指定环境是否支持处理 Web 请求的 Web 应用程序或运行后台作业的 Web 应用程序。有关更多信息，请参阅Elastic Beanstalk 工作线程环境。您可指定 `WebServer` 或 `Worker`。
- InstanceType – 指定具有最适合应用程序的特征（包括内存大小和 CPU 处理能力）的服务器。有关 Amazon EC2 实例系列和类型的更多信息，请参阅 Amazon EC2 用户指南中的实例类型或亚马逊 EC2 用户指南中的实例类型。有关各区域可用实例类型的更多信息，请参阅 Amazon EC2 用户指南中的可用实例类型或 Ama zon EC2 用户指南中的可用实例类型。
- withVpc – 指定是否在 Amazon VPC 中创建环境。您可指定 `true` 或 `false`。有关配合使用 Elastic Beanstalk 和 Amazon VPC 的更多信息，请参阅将 Elastic Beanstalk 和 Amazon VPC 结合使用。
- withRds – 指定是否使用此环境创建 Amazon RDS 数据库实例。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。您可指定 `true` 或 `false`。
- rdsDBEngine – 指定要在此环境中用于 Amazon EC2 实例的数据库引擎。您可指定 `mysql`、`oracle-sel`、`sqlserver-ex`、`sqlserver-web` 或 `sqlserver-se`。默认值为 `mysql`。
- rdsDB AllocatedStorage-以千兆字节 (GB) 为单位指定分配的数据库存储大小。可以指定以下值：
  - MySQL – 5 到 1024。默认为 5。
  - Oracle – 10 到 1024。默认为 10。
  - Microsoft SQL Server Express Edition – 30。
  - Microsoft SQL Server Web Edition – 30。
  - Microsoft SQL Server Standard Edition – 200。
- RdsDB InstanceClass-指定数据库实例类型。默认值为 `db.t2.micro`（`db.m1.large` 适用于未在 Amazon VPC 中运行的环境）。有关 Amazon RDS 支持的数据库实例类的列表，请参阅《Amazon Relational Database Service 用户指南》中的数据库实例类。
- rdsMultiAZDatabase – 指定 Elastic Beanstalk 是否需要跨多个可用区创建数据库实例。您可指定 `true` 或 `false`。有关使用 Amazon RDS 进行多个可用区部署的更多信息，请参阅《Amazon Relational Database Service 用户指南》中的区域和可用区。

- RdsDB DeletionPolicy — 指定在环境终止时是删除数据库实例还是对数据库实例进行快照。您可指定 `Delete` 或 `Snapshot`。

示例

下面是一个示例 Launch Now URL。构建您自己的 URL 之后，您可以将它提供给您的用户。例如，您可以将该 URL 嵌入在网页或培训材料中。当用户使用 Launch Now URL 创建应用程序时，Elastic Beanstalk 的“创建应用程序”向导不需要其他输入。

```
https://console.aws.amazon.com/elasticbeanstalk/home?region=us-west-2#/
newApplication?applicationName=YourCompanySampleApp&platform=PHP%207.3%20running
%20on%2064bit%20Amazon%20Linux&sourceBundleUrl=http://s3.amazonaws.com/mybucket/
myobject&environmentType=SingleInstance&tierName=WebServer&instanceType=m1.small&withVpc=true&w
```

当用户选择 Launch Now URL 时，Elastic Beanstalk 会显示如下页面。

Create a web app

Create a new application and environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. Learn more

Application information

Application name YourCompanySampleApp

Up to 100 Unicode characters, not including forward slash (/).

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Environment name Yourcompanysampleapp-env

Domain Leave blank for autogenerated value .us-east-1.elasticbeanstalk.com

Check availability

Description

Base configuration

Tier Web Server (Choose tier)

Platform Preconfigured platform

Platforms published and maintained by AWS Elastic Beanstalk.

PHP

Custom platform NEW

Platforms created and owned by you. Learn more

-- Choose a custom platform --

Application code Sample application

Get started right away with sample code.

Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Upload ZIP or WAR

Cancel Review and launch

使用 Launch Now URL

1. 选择 Launch Now URL。
2. 当 Elastic Beanstalk 控制台打开后，在 Create a web app（创建 Web 应用程序）页面上，选择 Review and launch（审核和启动）以查看 Elastic Beanstalk 用于创建应用程序和启动应用程序运行环境的设置。
3. 在配置页面上，选择创建应用程序以创建应用程序。

创建和更新 Elastic Beanstalk 环境组

借助 AWS Elastic Beanstalk `Compose Environments` API，您可以在单个应用程序中创建和更新 Elastic Beanstalk 环境组。组中的每个环境都可以运行服务导向型架构应用程序的一个独立组件。`Compose Environments` API 使用一个应用程序版本列表和一个可选组名。Elastic Beanstalk 为每个应用程序版本创建一个环境，如果环境已经存在，则将应用程序版本部署到相应环境。

在 Elastic Beanstalk 环境之间创建链接，将一个环境指定为另一个环境的依赖项。当您使用 `Compose Environments` API 创建一组环境时，Elastic Beanstalk 仅在依赖关系启动并运行之后才会创建依赖环境。有关环境链接的更多信息，请参阅在 Elastic Beanstalk 环境之间创建链接。

`Compose Environments` API 使用环境清单存储由各个环境组共享的配置详细信息。每个组件应用程序在其应用程序源包中都必须有一个 `env.yaml` 配置文件，用于指定创建它的环境时所使用的参数。

`Compose Environments` 要求在每个组件应用程序的环境清单中指定 `EnvironmentName` 和 `SolutionStack`。

您可以将 `Compose Environments` API 与 Elastic Beanstalk 命令行接口 (EB CLI) AWS CLI、或 SDK 一起使用。有关 EB CLI 说明，请参阅使用 EB CLI 以组的形式管理多个 Elastic Beanstalk 环境。

使用 **`Compose Environments`** API

例如，您可以创建一个名为 `Media Library` 的应用程序，让用户通过它上传和管理 Amazon Simple Storage Service (Amazon S3) 中存储的图像和视频。该应用程序的前端环境 `front` 运行一个 Web 应用程序，用户可通过该 Web 应用程序上传和下载各个文件、查看库和启动批处理作业。

前端应用程序不直接处理作业，而是将作业添加到 Amazon SQS 队列中。第二个环境 `worker` 从队列中提取作业并进行处理。`worker` 使用一个具有高性能 GPU 的 G2 实例类型，而 `front` 可以在更为经济实用的通用实例类型上运行。

您可以通过组织项目文件夹 Media Library 为每个组件使用单独的目录，每个目录各在一个环境定义文件 (env.yaml) 中包含相应组件的源代码：

```
~/workspace/media-library
|-- front
|   `-- env.yaml
`-- worker
    `-- env.yaml
```

以下列表显示了各个组件应用程序的 env.yaml 文件。

~/workspace/media-library/front/env.yaml

```
EnvironmentName: front+
EnvironmentLinks:
  "WORKERQUEUE" : "worker+"
AWSConfigurationTemplateVersion: 1.1.0.0
EnvironmentTier:
  Name: WebServer
  Type: Standard
SolutionStack: 64bit Amazon Linux 2015.09 v2.0.4 running Java 8
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: m4.large
```

~/workspace/media-library/worker/env.yaml

```
EnvironmentName: worker+
AWSConfigurationTemplateVersion: 1.1.0.0
EnvironmentTier:
  Name: Worker
  Type: SQS/HTTP
SolutionStack: 64bit Amazon Linux 2015.09 v2.0.4 running Java 8
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: g2.2xlarge
```

在为前端 (front-v1) 和工作程序 (worker-v1) 应用程序组件创建应用程序版本后，您可以使用版本名调用 Compose Environments API。在此示例中，我们使用 AWS CLI 调用 API。

```
# Create application versions for each component:
```

```
~$ aws elasticbeanstalk create-application-version --application-name media-
library --version-label front-v1 --process --source-bundle S3Bucket="DOC-EXAMPLE-
BUCKET",S3Key="front-v1.zip"
  {
    "ApplicationVersion": {
        "ApplicationName": "media-library",
        "VersionLabel": "front-v1",
        "Description": "",
        "DateCreated": "2015-11-03T23:01:25.412Z",
        "DateUpdated": "2015-11-03T23:01:25.412Z",
        "SourceBundle": {
            "S3Bucket": "DOC-EXAMPLE-BUCKET",
            "S3Key": "front-v1.zip"
        }
    }
  }
~$ aws elasticbeanstalk create-application-version --application-name media-
library --version-label worker-v1 --process --source-bundle S3Bucket="DOC-EXAMPLE-
BUCKET",S3Key="worker-v1.zip"
  {
    "ApplicationVersion": {
        "ApplicationName": "media-library",
        "VersionLabel": "worker-v1",
        "Description": "",
        "DateCreated": "2015-11-03T23:01:48.151Z",
        "DateUpdated": "2015-11-03T23:01:48.151Z",
        "SourceBundle": {
            "S3Bucket": "DOC-EXAMPLE-BUCKET",
            "S3Key": "worker-v1.zip"
        }
    }
  }
# Create environments:
~$ aws elasticbeanstalk compose-environments --application-name media-library --group-
name dev --version-labels front-v1 worker-v1
```

第三个调用创建两个环境：front-dev 和 worker-dev。该 API 通过将 EnvironmentName 文件中指定的 env.yaml 与 group name 调用中指定的 Compose Environments 选项用连字符连接起来创建环境名。这两个选项加连字符的总长度不得超过 23 个字符的环境名最大长度限制。

在 front-dev 环境中运行的应用程序可以通过读取 worker-dev 变量来访问附加到 WORKERQUEUE 环境的 Amazon SQS 队列的名称。有关环境链接的更多信息，请参阅在 Elastic Beanstalk 环境之间创建链接。

将应用程序部署到 Elastic Beanstalk 环境

您可以使用 AWS Elastic Beanstalk 控制台上传更新的源代码包并将其部署到 Elastic Beanstalk 环境中，或者重新部署以前上传的版本。

每项部署均由部署 ID 标识。部署 ID 从 1 开始，每次部署和更改实例配置时，部署 ID 就会逐一往上增加。如果您启用了增强型运行状况报告，则 Elastic Beanstalk 会在报告实例运行状况时，同时在运行状况控制台和 EB CLI 中显示部署 ID。滚动更新失败时，部署 ID 帮助您确定环境的状态。

Elastic Beanstalk 提供了多种部署策略和设置。有关配置策略和附加设置的详细信息，请参阅the section called “部署选项”。下表列出了支持这些设置的策略和环境类型。

支持的部署策略

部署策略	负载均衡环境	单实例环境	传统 Windows Server 环境†
一次部署全部	✓ 是	✓ 是	✓ 是
滚动	✓ 是	X 否	✓ 是
附加批次滚动部署	✓ 是	X 否	X 否
不可变的	✓ 是	✓ 是	X 否
流量拆分	✓ 是 (Application Load Balancer)	X 否	X 否

† 在此表中，传统 Windows Server 环境 是一个基于 Windows Server 平台配置的环境，所使用的 IIS 版本早于 IIS 8.5。

> **Warning**
>
> 某些策略会在部署或更新期间替换所有实例。这会导致丢失所有累积的 Amazon EC2 突发余额。这发生在以下情况下：
>
> - 已启用实例替换的托管平台更新
> - 不可变更新
> - 已启用不可变更新或流量拆分的部署

选择部署策略

为您的应用程序选择正确的部署策略时，需要权衡一些考虑因素，并且根据您的特定需求而定。the section called “部署选项”页面包含有关每个策略的更多信息，以及其中一些策略的工作方式的详细说明。

以下列表提供了有关不同部署策略的概要信息，并增加了相关注意事项。

- 一次部署全部 – 最快的部署方法。如果您可以接受短期服务停机，并且快速部署对您非常重要，则这种方式适合您。使用此方法，Elastic Beanstalk 会将新的应用程序版本部署到每个实例。然后，可能需要重新启动 Web 代理或应用程序服务器。因此，用户可能会在短时间内无法使用您的应用程序（或可用性很低）。
- 滚动部署 – 可避免停机时间，并最大限度地减少对可用性的影响，但代价是部署时间会比较长。如果您无法接受服务在任何时间内完全停机，则这种方式适合您。使用此方法，您的应用程序一次会将一批实例部署到您的环境中。在整个部署过程中，大多数带宽都会保留。
- 附加批次滚动部署 – 可避免任何可用性降低的情况，代价是与滚动部署 方法相比，部署时间甚至会更长。适用于在整个部署过程中必须保留相同带宽的情况。使用此方法，Elastic Beanstalk 会启动额外的实例批次，然后执行滚动部署。启动额外的批次需要花费时间，并确保在整个部署过程中保留相同的带宽。
- 不可变的 – 一种较慢的部署方法，可确保您的新应用程序版本始终部署到新实例，而不是更新现有实例。它还有在部署失败时可快速安全回滚的额外优势。使用此方法，Elastic Beanstalk 会执行不可变的更新来部署您的应用程序。在不可变更新中，环境中将启动第二个 Auto Scaling 组，新版本与旧版本一起为流量提供服务，直至新实例通过运行状况检查。
- 流量拆分 – 一种 Canary 测试部署方法。如果您希望使用传入流量的一部分来测试新应用程序版本的运行状况，同时保留旧应用程序版本提供的其余流量，则这种方式适合您。

下表比较了部署方法属性。

部署方法

方法	部署失败带来的影响	部署时间	零停机时间	无 DNS 更改	回滚过程	代码部署到
一次部署全部	停机时间	🕒	X 否	✓ 是	手动重新部署	现有实例

方法	部署失败带来的影响	部署时间	零停机时间	无 DNS 更改	回滚过程	代码部署到
滚动	单个批次服务中断；任何在故障之前成功的批次将运行新应用程序版本	🕒	✓ 是	✓ 是	手动重新部署	现有实例
附加批次滚动部署	如果第一个批次失败，则影响最小；否则类似于滚动	🕒	✓ 是	✓ 是	手动重新部署	新实例和现有实例
不可变的	最低	🕒	✓ 是	✓ 是	终止新实例	新实例
流量拆分	路由到新版本的客户端流量的百分比会临时受到影响	🕒	✓ 是	✓ 是	重新路由流量并终止新实例	新实例
蓝/绿	最低	🕒	✓ 是	X 否	交换 URL	新实例

† 根据批次大小而变化。

†† 因评估时间选项设置而异。

部署新应用程序版本

您可以从环境的控制面板执行部署。

将新应用程序版本部署到 Elastic Beanstalk 环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的表单上传应用程序源代码包。
5. 选择 Deploy（部署）。

重新部署先前版本

您也可以从应用程序版本页面将之前上传的应用程序版本部署到其任一环境。

将现有的应用程序版本部署到现有的环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Application versions（应用程序版本）。
4. 选择要部署的应用程序版本。
5. 选择 Actions（操作），然后选择 Deploy（部署）。
6. 选择环境，然后选择 Deploy（部署）。

部署您的应用程序的其他方法

如果经常部署，可以考虑使用 Elastic Beanstalk 命令行界面（EB CLI）管理您的环境。EB CLI 将在随源代码一起创建一个存储库。它还可以创建源代码包，将其上传到 Elastic Beanstalk，然后使用单个命令部署该源代码包。

针对依赖于资源配置更改的部署，或者不能与旧版本一起运行的新版本的部署，您可以使用新版本启动新环境，然后为蓝/绿部署执行别名记录交换。

部署策略和设置

AWS Elastic Beanstalk 提供了多个有关如何处理部署的选项，包括部署策略（一次部署全部、滚动部署、额外批量滚动、不可变和流量拆分）以及可用于在部署期间配置批大小和运行状况检查行为的选项。默认情况下，您的环境使用一次性部署。如果您使用 EB CLI 创建了环境，并且它是可扩展的环境（您未指定 `--single` 选项），则它使用滚动部署。

通过滚动部署，Elastic Beanstalk 将环境的 Amazon EC2 实例拆分为各个批，并将新版本的应用程序一次部署到一个批中。运行旧版应用程序的环境中的其余实例将会保留。在滚动部署期间，一些实例通过旧版本的应用程序处理请求，而已完成批次中的实例通过新版本处理其他请求。有关详细信息，请参阅 the section called “滚动部署的工作方式”。

要在部署期间保持完整容量，可以配置环境以启动新的实例批次，然后再禁用任何实例。此选项称作附加批次滚动部署。在部署完成后，Elastic Beanstalk 将终止附加的实例批次。

不可变的部署执行不可变更新，以启动在单独的 Auto Scaling 组中运行新版本应用程序的一组完整的新实例，以及运行旧版本的实例。不可变的部署会阻止由部分完成的滚动部署导致的问题。如果新实例未通过运行状况检查，则 Elastic Beanstalk 将终止这些实例，并使原始实例保持不变。

流量拆分部署允许您在应用程序部署过程中执行 Canary 测试。在流量拆分部署中，Elastic Beanstalk 启动一整套新实例，就像在不可变部署期间一样。然后，它在指定的评估期内将指定百分比的传入客户端流量转发到新的应用程序版本。如果新实例保持正常状态，则 Elastic Beanstalk 会将所有流量转发给它们并终止旧实例。如果新实例未通过运行状况检查，或者您选择中止部署，则 Elastic Beanstalk 会将流量移回旧实例并终止新实例。从不会有任何服务中断。有关详细信息，请参阅 the section called “流量拆分部署的工作方式”。

> **Warning**
>
> 某些策略会在部署或更新期间替换所有实例。这会导致丢失所有累积的 Amazon EC2 突发余额。这发生在以下情况下：
>
> - 已启用实例替换的托管平台更新
> - 不可变更新
> - 已启用不可变更新或流量拆分的部署

如果您的应用程序未通过所有运行状况检查，但仍能够以较低级别的运行状况正常运行，则您可以修改正常阈值选项，使实例能够通过较低级别状况（例如 `Warning`）的运行状况检查。如果您的部署因

未通过运行状况检查而失败，并且您需要强制实施更新而不管运行状况如何，请指定忽略运行状况检查选项。

如果您指定滚动更新的批大小，则 Elastic Beanstalk 还将使用该值来滚动重新启动应用程序。在您需要重新启动运行于环境实例上的代理和应用程序服务器而不会导致停机时，可使用滚动重新启动。

配置应用程序部署

在环境管理控制台中，通过编辑环境的 Configuration（配置）页面上的 Updates and Deployments（更新和部署）来启用和配置批量应用程序版本部署。

配置部署 (控制台)

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Rolling updates and deployments（滚动更新和部署）配置类别中，选择 Edit（编辑）。
5. 在应用程序部署部分中，选择部署策略、批设置和运行状况检查选项。
6. 要保存更改，请选择页面底部的 Apply（应用）。

Rolling updates and deployments（滚动更新和部署）页面的 Application deployments（应用程序部署）部分提供以下应用程序部署选项：

- Deployment policy（部署策略）– 从以下部署选项中选择：
  - All at once（一次部署全部）– 将新版本同时部署到所有实例。在执行部署时，环境中的所有实例将短时间禁用。
  - Rolling（滚动）– 批量部署新版本。在部署阶段，每个批次都将禁用，以便按批次中的实例数减少环境的容量。
  - Rolling with additional batch（额外批量滚动）– 批量部署新版本，但首先启动新的实例批次以确保部署过程中能够使用完整容量。
  - Immutable（不可变的）– 通过执行不可变更新，将新版本部署到新实例组。

- Traffic splitting（流量拆分）– 将新版本部署到一组新的实例，并在现有应用程序版本与新版本之间临时拆分传入的客户端流量。

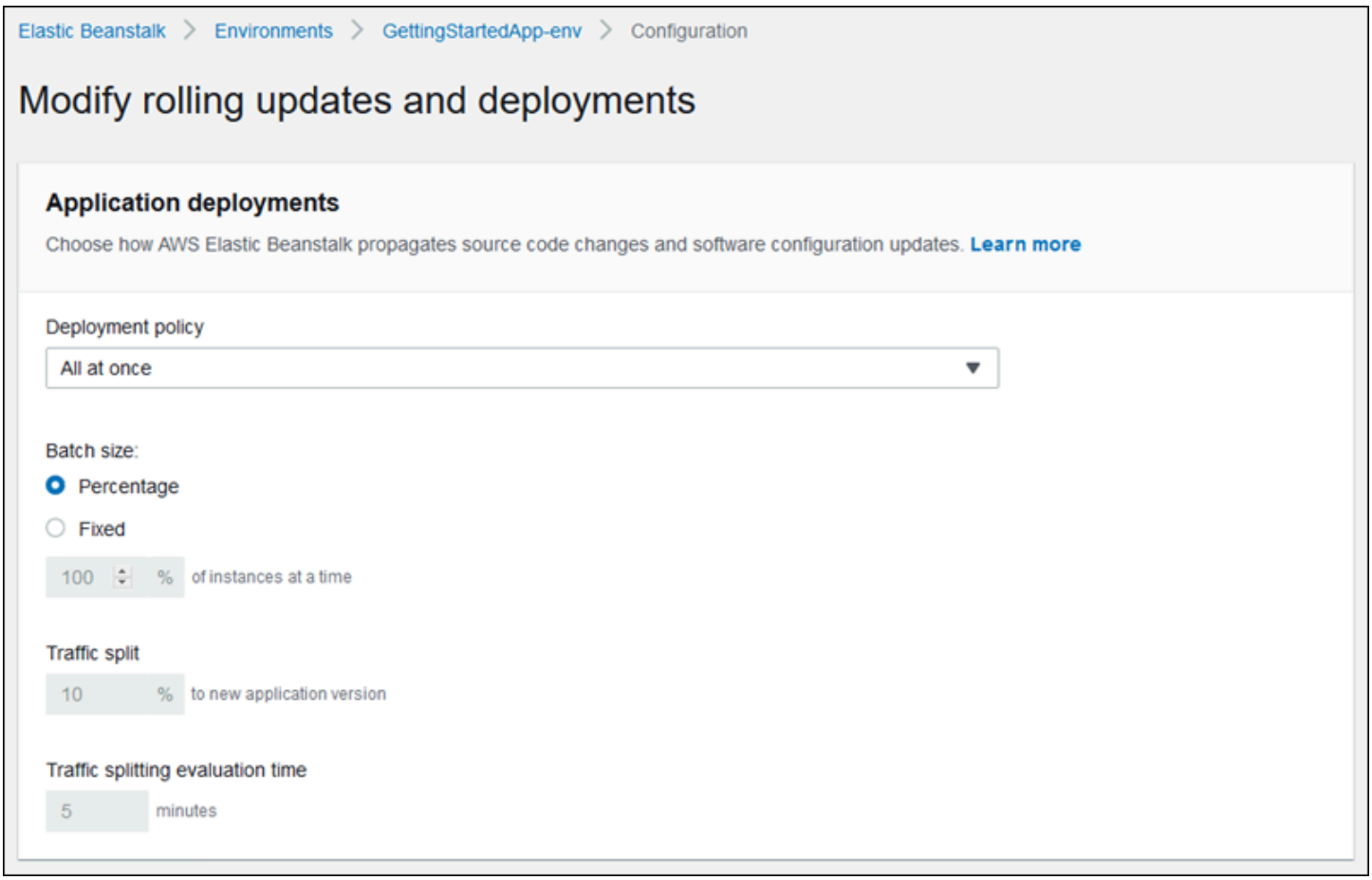
对于滚动部署和附加批次滚动部署部署策略，您可以配置：

- Batch size（批处理大小）– 每个批次中要部署的实例集大小。

  选择 Percentage（百分比）来配置占 Auto Scaling 组中的 EC2 实例总数的百分比（最高 100%），或选择 Fixed（固定）来配置固定数量的实例（最高您的环境的 Auto Scaling 配置中的最大实例计数）。

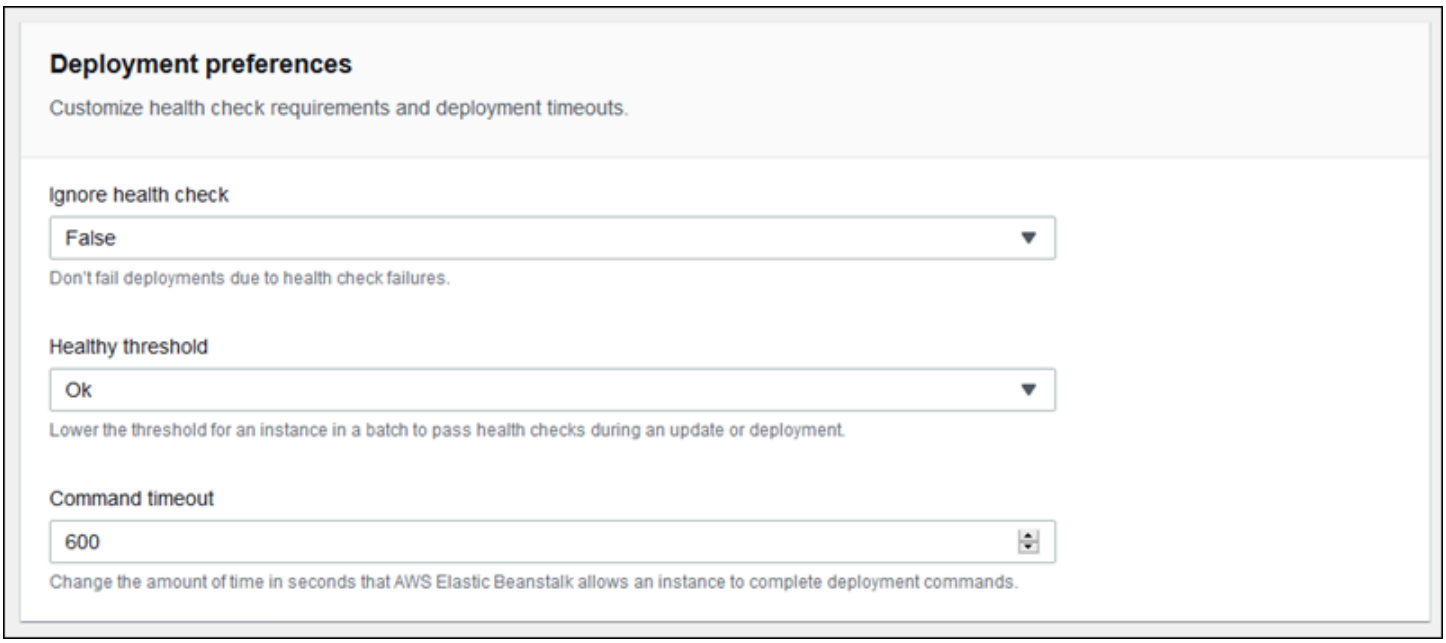
对于流量拆分部署策略，您可以配置以下内容：

- Traffic split（流量拆分）– 对于运行您正在部署的新应用程序版本的环境实例，Elastic Beanstalk 转移到这些实例的传入客户端流量的初始百分比。
- 流量拆分评估时间 – 在初始正常部署之后，Elastic Beanstalk 等待的时间段（以分钟为单位），在经过该时间后，会继续将所有传入的客户端流量转移到正部署的新应用程序版本。



部署首选项部分包含与运行状况检查相关的选项。

- Ignore health check（忽略运行状况检查）– 当某个批次在 Command timeout（命令超时）内未能达到良好的运行状况时，阻止部署回滚。
- Healthy threshold（正常阈值）– 减小在滚动部署、滚动更新和不可变更新期间将实例视为良好运行的阈值。
- Command timeout（命令超时）– 在取消部署或继续下一批次 [如果设置 Ignore health check（忽略运行状况检查）前等待实例达到良好运行状况的时间（以秒为单位）。



滚动部署的工作方式

在处理某个批次时，Elastic Beanstalk 会将该批次中的所有实例从负载均衡器中分离，部署新的应用程序版本，然后重新挂载这些实例。如果您已启用 Connection Draining，则 Elastic Beanstalk 将在开始部署之前耗尽来自每个批次中的 Amazon EC2 实例的现有连接。

在某个批次中的实例重新挂载到负载均衡器之后，Elastic Load Balancing 将等到它们通过最小数量的 Elastic Load Balancing运行状况检查（Healthy check count threshold（运行状况检查计数阈值）],然后开始将流量路由到它们。如果未配置运行状况检查 URL，这可以很快发生，因为实例一旦能接受 TCP 连接便会立即通过运行状况检查。如果配置了运行状况检查 URL，则在更新后的实例返回 200 OK 状态代码来响应对运行状况检查 URL 的 HTTP GET 请求之前，负载均衡器不会将流量路由到这些实例。

Elastic Beanstalk 将等到一个批次中的所有实例的运行状况正常之后才会转到下一个批次。当采用基本运行状况报告时，实例运行状况取决于 Elastic Load Balancing 运行状况检查状态。当某个批次中的所有实例均已通过 Elastic Load Balancing 认为足够数量的运行状况检查之后，该批次便已完成。如果启用增强型运行状况报告，则 Elastic Beanstalk 需考虑几个其他因素，包括传入请求的结果。当采用增强型运行状况报告时，对于 Web 服务器环境，所有实例都必须在 2 分钟内通过 12 次连续的运行状况检查并呈现正常状态；对于工作线程环境，则必须在 3 分钟内通过 18 次运行状况检查。

如果一批实例在命令超时范围内未转变为运行状况良好状态，则表示部署失败。在部署失败后，请检查环境中实例的运行状况以了解有关失败原因的信息。然后使用固定或已知良好版本的应用程序再次执行部署以进行回滚。

如果在一批或多批实例成功完成后部署失败，则已完成的批次将运行新版本的应用程序，而待处理的批次则会继续运行旧版本。您可以通过控制台中的运行状况页面识别在环境中的实例上运行的版本。此页面会显示在环境中的每个实例上执行的最新部署的部署 ID。如果您终止失败部署中的实例，Elastic Beanstalk 将从最新的成功部署中选择运行应用程序版本的实例来替换它们。

流量拆分部署的工作方式

使用流量拆分部署，您可以执行 Canary 测试。在提交到新版本并将所有流量定向到新的应用程序版本之前，您可以将一些传入的客户端流量定向到新的应用程序版本，以便验证应用程序的运行状况。

在流量拆分部署期间，Elastic Beanstalk 在单独的临时 Auto Scaling 组中创建一组新实例。然后，Elastic Beanstalk 指示负载均衡器将您环境中一定百分比的传入流量定向到新实例。接下来，在配置的时间段内，Elastic Beanstalk 会跟踪新实例集的运行状况。如果一切正常，则 Elastic Beanstalk 会将剩余流量转移到新实例，并将这些实例附加到环境的原始 Auto Scaling 组，以便替换旧实例。然后 Elastic Beanstalk 进行清理，即终止旧实例并删除临时 Auto Scaling 组。

> **Note**
>
> 在流量拆分部署期间，环境的容量不会发生变化。Elastic Beanstalk 在临时 Auto Scaling 组中启动与部署开始时原始 Auto Scaling 组中相同数量的实例。然后，在部署持续时间内，它会在两个 Auto Scaling 组中维护一定数量的实例。在配置环境的流量拆分评估时间时，请考虑这一事实。

将部署回滚到以前的应用程序版本的速度非常快，不会影响服务到客户端的流量。如果新实例未通过运行状况检查，或者您选择中止部署，则 Elastic Beanstalk 会将流量移回旧实例并终止新实例。通过使用 Elastic Beanstalk 控制台中的环境概述页面，并在 Environment actions（环境操作）中选择 Abort current operation（中止当前操作），可以中止任何部署。您还可以调用 AbortEnvironmentUpdate API 或等效的 AWS CLI 命令。

流量拆分部署需要 Application Load Balancer。在您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境时，Elastic Beanstalk 默认使用此负载均衡器类型。

部署选项命名空间

您可以使用 `aws:elasticbeanstalk:command` 命名空间中的配置选项来配置您的部署。如果您选择流量拆分策略，则此策略的附加选项会在 `aws:elasticbeanstalk:trafficsplitting` 命名空间中可用。

使用 DeploymentPolicy 选项可设置部署类型。支持下列值：

- AllAtOnce – 禁用滚动部署并且始终同时部署到所有实例。
- Rolling – 启用标准滚动部署。
- RollingWithAdditionalBatch – 在开始部署之前将启动额外的实例批次，以便保持完整容量。
- Immutable – 对每次部署执行不可变更新。
- TrafficSplitting – 执行流量拆分部署，以便对应用程序部署进行 Canary 测试。

启用滚动部署时，设置 BatchSize 和 BatchSizeType 选项以配置每个批次的大小。例如，要部署每个批次中 25% 的实例，请指定以下选项和值：

Example .ebextensions/rolling-updates.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Rolling
    BatchSizeType: Percentage
    BatchSize: 25
```

要部署到每个批次中的五个实例（不管运行的实例数量如何），并且在将任何实例从服务中拉出之前启动额外的批次（五个实例）来运行新版本，请指定以下选项和值。

Example .ebextensions/rolling-additionalbatch.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: RollingWithAdditionalBatch
    BatchSizeType: Fixed
    BatchSize: 5
```

要对运行状况检查阈值为警告的每个部署执行不可变更新，并在批次中的实例未在 15 分钟超时时间内通过运行状况检查的情况下继续部署，请指定以下选项和值。

Example .ebextensions/immutable-ignorehealth.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Immutable
```

```
    HealthCheckSuccessThreshold: Warning
    IgnoreHealthCheck: true
    Timeout: "900"
```

要执行流量拆分部署、将 15% 的客户端流量转发到新应用程序版本并评估运行状况 10 分钟时间，请指定以下选项和值。

Example .ebextensions/traffic-splitting.config

```
option_settings:
  aws:elasticbeanstalk:command:
    DeploymentPolicy: TrafficSplitting
  aws:elasticbeanstalk:trafficsplitting:
    NewVersionPercent: "15"
    EvaluationTime: "10"
```

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

使用 Elastic Beanstalk 进行蓝/绿部署

因为 AWS Elastic Beanstalk 在您更新应用程序版本时执行就地更新，所以应用程序可能会对用户短时间不可用。为避免这种情况，请执行蓝/绿部署。如要执行此操作，将新版本部署到独立的环境，然后交换两个环境的别名记录，从而将流量立即重新导向到新版本。

如果您希望将环境更新为不兼容的平台版本时，还需要蓝/绿部署。有关更多信息，请参阅the section called “平台更新”。

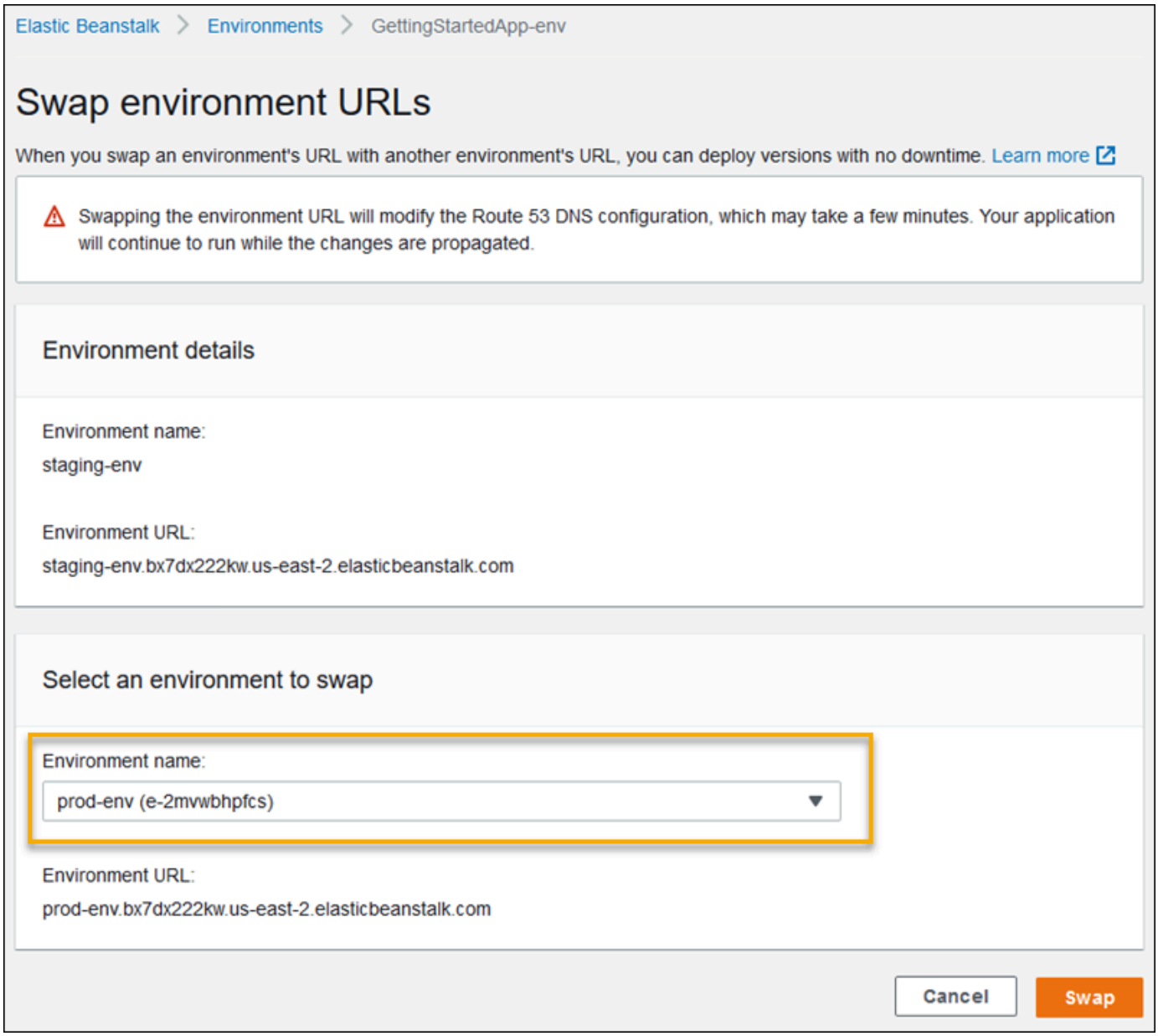
当应用程序使用生产数据库时，蓝/绿部署要求您的环境独立于生产数据库运行。如果您的环境中包含 Elastic Beanstalk 代表您创建的数据库，除非您执行特定操作，否则系统不会保留该环境的数据库和连接。如果您有想要保留的数据库，请使用以下 Elastic Beanstalk 数据库生命周期选项之一。在解耦数据库后，可以选择 Retain（保留）选项以保持数据库和环境的运行状态。有关更多信息，请参阅本指南配置环境章节中的数据库生命周期。

有关如何将您的应用程序配置为连接到 Amazon RDS 实例（非 Elastic Beanstalk 托管）的说明，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

执行蓝/绿部署

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。

2. 克隆当前环境，或启动运行所需平台版本的新环境。
3. 向新环境部署新应用程序版本。
4. 在新环境上测试新版本。
5. 在环境概述页面上，选择 Actions（操作），然后选择 Swap environment URLs（交换环境 URL）。
6. 对于 Environment name（环境名称），请选择当前环境。



7. 选择 Swap（交换）。

Elastic Beanstalk 交换旧环境和新环境的别名记录，以将流量从旧版本重定向到新版本。

Elastic Beanstalk 完成交换操作之后，验证在您尝试连接到旧环境 URL 时新环境是否响应。但是，在传播 DNS 更改并且您的旧 DNS 记录过期之前，请勿终止旧环境。DNS 服务器不总是会基于您在 DNS 记录中设置的存活时间（TTL）来清除其缓存中的旧记录。

配置更改

当您在环境管理控制台的 Configuration 部分修改配置选项设置时，AWS Elastic Beanstalk 将更改传播到所有受影响的资源。这些资源包括将流量分配到运行您的应用程序的 Amazon EC2 实例的负载均衡器、管理这些实例的 Auto Scaling 组以及 EC2 实例本身。

很多配置更改都可应用于正在运行的环境而不用替换现有实例。例如，设置运行状况检查 URL 将触发环境更新以修改负载均衡器设置，但不会导致任何停机时间，因为运行您的应用程序的实例在更新传播时会继续处理请求。

修改启动配置或 VPC 设置的配置更改需要终止您的环境中的所有实例并替换它们。例如，当您更改环境的实例类型或 SSH 密钥设置时，必须终止和替换您的 EC2 实例。Elastic Beanstalk 提供了多项策略来确定如何执行此更换。

- 滚动更新 - Elastic Beanstalk 将批量应用这些配置更改，让最小数量的实例保持运行并始终提供流量服务。此方法可防止更新过程中的停机。有关详细信息，请参阅滚动更新。
- 不可变更新 - Elastic Beanstalk 在您的环境外启动一个临时 Auto Scaling 组，其中一组单独的实例使用新配置运行。然后，Elastic Beanstalk 将这些实例放在环境的负载均衡器后面。旧实例和新实例均为流量提供服务，直至新实例通过运行状况检查。届时，Elastic Beanstalk 将新实例移动到环境的 Auto Scaling 组中，并终止临时组和旧实例。有关详细信息，请参阅不可变更新。
- 禁用 - Elastic Beanstalk 不会尝试避免停机。它终止环境的现有实例，并将其替换为使用新配置运行的新实例。

> **Warning**
>
> 某些策略会在部署或更新期间替换所有实例。这会导致丢失所有累积的 Amazon EC2 突发余额。这发生在以下情况下：
>
> - 已启用实例替换的托管平台更新
> - 不可变更新
> - 已启用不可变更新或流量拆分的部署

支持的更新类型

滚动更新设置	负载均衡环境	单实例环境	传统 Windows Server 环境†
已禁用	✓是	✓是	✓是
根据运行状况滚动	✓是	×否	✓是
基于时间滚动	✓是	×否	✓是
不可变的	✓是	✓是	×否

† 针对此表的用途，传统 Windows Server 环境 是一个基于 Windows Server 平台配置的环境，所使用的 IIS 版本早于 IIS 8.5。

主题

- Elastic Beanstalk 滚动环境配置更新
- 不可变的环境更新

Elastic Beanstalk 滚动环境配置更新

当配置更改要求替换实例时，Elastic Beanstalk 可以批量执行更新以避免在传播更改时停机。在滚动更新期间，容量只按单批次大小减少，而您可以配置这个大小。Elastic Beanstalk 禁用一批实例，终止这些实例，然后启动一批具有新配置的实例。在新批次开始处理请求后，Elastic Beanstalk 将移至下一个批次。

滚动配置更新批次可以基于时间定期处理 (每个批次之间有延迟) 或根据运行状况来处理。对于基于时间的滚动更新，您可以配置 Elastic Beanstalk 在启动完一批实例后和移至下一个批次前所等待的时间。此暂停时间可让您的应用程序引导并开始处理请求。

当采用基于运行状况的滚动更新时，Elastic Beanstalk 会等一个批次中的实例通过运行状况检查之后再继续下一个批次。实例的运行状况由运行状况报告系统确定，可以是基本运行状况或增强型运行状况。当采用基本运行状况时，一旦某个批次中的所有实例都通过 Elastic Load Balancing (ELB) 运行状况检查，该批次即被视为运行状况良好。

当采用增强型运行状况报告时，某个批次中的所有实例必须先通过连续的多次运行状况检查，Elastic Beanstalk 才能移至下一个批次。除了 ELB 运行状况检查（仅检查您的实例）之外，增强型运行状况

还监视应用程序日志和环境中其他资源的状态。在采用增强型运行状况报告的 Web 服务器环境中，所有实例都必须在 2 分钟内通过 12 项运行状况检查 (对于工作线程环境，则必须在 12 分钟内通过 18 次检查)。如果任何实例未能通过一项运行状况检查，则将重置计数。

如果某个批次在滚动更新超时时间内（默认为 30 分钟）未变成正常运行，则会取消更新。滚动更新超时时间是一个在 `aws:autoscaling:updatepolicy:rollingupdate` 命名空间中可用的配置选项。如果应用程序未通过运行状况检查（即未处于 Ok 状态），但在另一级别很稳定，您可以在 `aws:elasticbeanstalk:healthreporting:system` 命名空间中设置 `HealthCheckSuccessThreshold` 选项，从而更改 Elastic Beanstalk 将实例视为运行状况良好的级别。

如果滚动更新过程失败，Elastic Beanstalk 会启动另一个滚动更新以回滚到之前的配置。运行状况检查失败或启动新实例导致超出账户配额都可能导致滚动更新失败。例如，如果达到了 Amazon EC2 实例数量的配额，在尝试配置一批新实例时滚动更新可能失败。在这种情况下，回滚也会失败。

回滚失败会终止更新过程，并令您的环境运行状况不佳。未处理的批次仍然使用旧的配置运行实例，而成功完成的批次具有新配置。要在回滚失败后修复环境，首先要解决导致更新失败的潜在问题，然后再次启动环境更新。

替代方法是将应用程序的新版本部署到不同的环境，然后执行别名记录交换重定向流量，这样不会产生停机时间。参阅 使用 Elastic Beanstalk 进行蓝/绿部署 了解更多信息。

滚动更新与滚动部署

如果更改需要为环境配置新 Amazon EC2 实例的设置，则会发生滚动更新。这包括对 Auto Scaling 组配置的更改（如实例类型和密钥对设置）和对 VPC 设置的更改。在滚动更新中，在配置新批次替换每个旧批次前，都会终止旧实例批次。

只要您部署应用程序就会发生滚动部署，通常无需替换环境中的实例就可以执行滚动部署。Elastic Beanstalk 将每个批次取出服务，部署新的应用程序版本，然后将其放回服务中。

例外情况是，在部署新应用程序版本的同时更改需要替换实例的设置。例如，如果您更改源包中的配置文件中的密钥名称设置并将其部署到环境，就会触发滚动更新。它不是将新应用程序版本部署到现有实例的每个批次，而是用新配置对新的实例批次进行配置。在这种情况下，不会发生单独部署，因为新应用程序版本采用新实例。

只要将新实例作为环境更新的一部分进行预配置，就有一个部署阶段，在此阶段，应用程序的源代码将部署到新实例，并且将应用任何修改实例上的操作系统或软件的配置设置。部署运行状况检查设置（忽略运行状况检查、正常阈值和命令超时）还在部署阶段应用于基于运行状况的滚动更新和不可变更新。

配置滚动更新

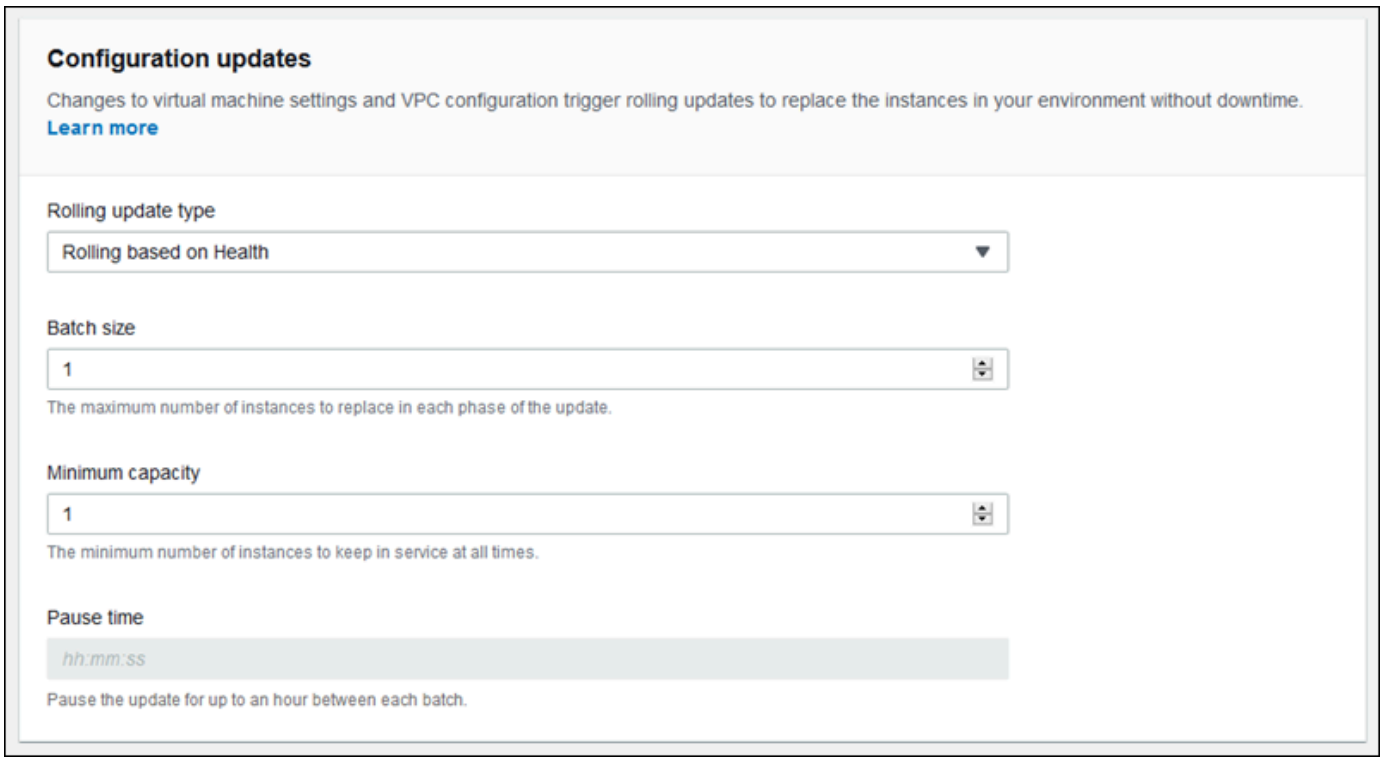
您可以在 Elastic Beanstalk 控制台中启用和配置滚动更新。

启用滚动更新

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Rolling updates and deployments（滚动更新和部署）配置类别中，选择 Edit（编辑）。
5. 在配置更新部分中，对于滚动更新类型，选择一个滚动选项。



6. 选择批处理大小、最小容量和暂停时间设置。
7. 要保存更改，请选择页面底部的 Apply（应用）。

Rolling updates and deployments（滚动更新和部署）页面的 Configuration updates（配置更新）部分提供以下滚动更新选项：

- Rolling update type (滚动更新类型) - Elastic Beanstalk 在完成一个实例批次更新后等待多长时间再进行下一批次，以允许这些实例完成引导启动并开始提供流量服务。从以下选项中进行选择：
  - Rolling based on Health (根据运行状况滚动) - 等待直至当前批次中的实例运行状况良好，然后再将这些实例投入使用并开始下一个批次。
  - Rolling based on Time (基于时间滚动) - 指定在开始下一个批次之前，在启动新实例和将实例投入使用之间等待的时间。
  - Immutable (不可变的) - 通过执行不可变更新将配置更改应用于新实例组。
- Batch size (批处理大小) - 每个批次要替换的实例的数量，介于 **1** 和 **10000** 之间。默认情况下，此值是 Auto Scaling 组的最小大小的三分之一，向上舍入取整。
- Minimum capacity (最小容量) - 在更新其他实例时要保持运行状态的最小实例数，介于 **0** 和 **9999** 之间。默认值为 Auto Scaling 组的最小大小，或 Auto Scaling 组的最大大小减一（取两者中较小的值）。
- Pause time (暂停时间)（仅限基于时间的滚动更新）- 更新一个批次后到继续处理下一批次前等待的时间，以允许您的应用程序开始接收流量。介于 0 秒到一小时之间。

aws:autoscaling:updatepolicy:rollingupdate 命名空间

您也可以使用 `aws:autoscaling:updatepolicy:rollingupdate` 命名空间中的配置选项来配置滚动更新。

使用 `RollingUpdateEnabled` 选项可启用滚动更新；使用 `RollingUpdateType` 可选择更新类型。`RollingUpdateType` 支持下列值：

- `Health` - 等待直至当前批次中的实例运行状况良好，然后再将这些实例投入使用并开始下一个批次。
- `Time` - 指定在开始下一个批次之前，在启动新实例和将实例投入使用之间等待的时间。
- `Immutable` - 通过执行不可变更新将配置更改应用于新实例组。

启用滚动更新时，设置 `MaxBatchSize` 和 `MinInstancesInService` 选项以配置每个批次的大小。对于基于时间的滚动更新和基于运行状况的滚动更新，您也可以分别配置 `PauseTime` 和 `Timeout`。

例如，要一次性启动最多 5 个实例并使至少 2 个实例保持使用状态，同时在各个批次之间等待 5 分 30 秒的时间，请指定以下选项和值。

Example .ebextensions/timebased.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateEnabled: true
    MaxBatchSize: 5
    MinInstancesInService: 2
    RollingUpdateType: Time
    PauseTime: PT5M30S
```

要启用基于运行状况的滚动更新，并将每个批次的超时设置为 45 分钟，请指定以下选项和值。

Example .ebextensions/healthbased.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateEnabled: true
    MaxBatchSize: 5
    MinInstancesInService: 2
    RollingUpdateType: Health
    Timeout: PT45M
```

`Timeout` 和 `PauseTime` 值必须以 ISO8601 持续时间格式指定：`PT#H#M#S`，其中各 `#` 分别是小时数、分钟数或秒数。

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

不可变的环境更新

不可变环境更新是滚动更新的替代方法。不可变环境更新可确保安全高效地应用需要替换实例的配置更改。如果不可变环境更新失败，则回滚过程仅需要终止 Auto Scaling 组。另一方面，一次失败的滚动更新需要执行额外的滚动更新来回滚更改。

为了执行不可变环境更新，Elastic Beanstalk 在您环境的负载均衡器后面创建第二个临时 Auto Scaling 组以包含新实例。首先，Elastic Beanstalk 在新组中使用新配置启动单个实例。此实例与在之前配置中运行的原始 Auto Scaling 组中的所有实例一起，为流量提供服务。

第一个实例通过了运行状况检查后，Elastic Beanstalk 使用新配置启动额外的实例，这些实例与在原始 Auto Scaling 组中运行的实例数相匹配。当所有新实例均通过运行状况检查后，Elastic Beanstalk 会将这些实例传输到原始 Auto Scaling 组，并终止临时 Auto Scaling 组和旧实例。

> **Note**
>
> 在不可变环境更新期间，当新 Auto Scaling 组中的实例开始处理请求并在终止原始 Auto Scaling 组的实例之前，您的环境在短时间内容量加倍。如果您的环境有许多实例，或者按需实例配额较低，请确保您有足够的容量来执行不可变的环境更新。如果您已接近配额，请考虑改为使用滚动更新。

不可变更新需要增强型运行状况报告来评估更新期间您环境的运行状况。增强型运行状况报告将标准负载均衡器运行状况检查与实例监视结合在一起，确保运行新配置的实例成功处理请求。

您还可以将不可变更新用作滚动部署的替代方法，用于部署应用程序的新版本。如果您配置 Elastic Beanstalk 来为应用程序部署使用不可变更新，在您每次部署应用程序的新版本时，它将替换您环境中的所有实例。如果不可变的应用程序部署失败，Elastic Beanstalk 将通过终止新 Auto Scaling 组来立即还原更改。这可以防止在某些批次完成后，在滚动部署失败时出现部分队列部署。

> **Warning**
>
> 某些策略会在部署或更新期间替换所有实例。这会导致丢失所有累积的 Amazon EC2 突发余额。这发生在以下情况下：
>
> - 已启用实例替换的托管平台更新
> - 不可变更新
> - 已启用不可变更新或流量拆分的部署

如果不可变更新失败，则新实例在 Elastic Beanstalk 终止它们之前，会将包日志上传到 Amazon S3。Elastic Beanstalk 将来自失败的不可变更新的日志在 Amazon S3 中保留一小时后才将其删除，而不是针对包日志和尾日志的标准 15 分钟。

> **Note**
>
> 如果您为应用程序版本部署使用不可变更新，但未对配置使用，则在您尝试部署的应用程序版本包含通常会触发滚动更新的配置更改 (例如，更改实例类型的配置) 时，可能会遇到错误。为避免这一点，请在单独的更新中进行配置更改，或者为部署和配置更改同时配置不可变更新。

您不能与资源配置更改一起执行不可变更新。例如，您不能在更改需要实例替换的设置的同时更新其他设置，或者执行其配置文件会在源代码中更改配置设置或附加资源的不可变部署。如果您尝试更改资源设置（例如，负载均衡器设置）并同时执行不可变更新，Elastic Beanstalk 将返回错误。

如果资源配置更改不依赖于源代码更改或者实例配置，请在两个更新中执行它们。如果有依赖关系，请改为执行蓝/绿部署。

配置不可变更新

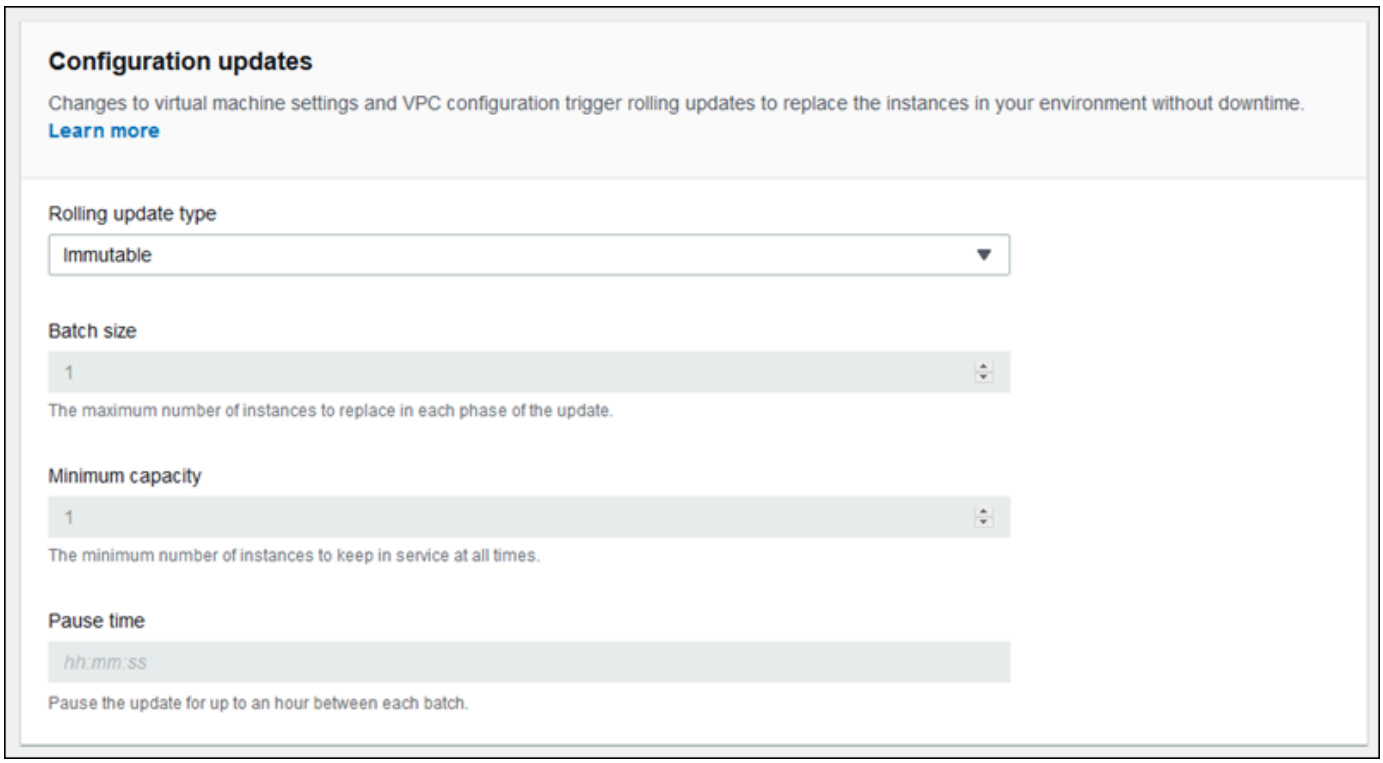
您可以在 Elastic Beanstalk 控制台中启用和配置不可变更新。

启用不可变更新 (控制台)

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Rolling updates and deployments（滚动更新和部署）配置类别中，选择 Edit（编辑）。
5. 在配置更新部分中，将滚动更新类型设置为不可变的。



6. 要保存更改，请选择页面底部的 Apply（应用）。

aws:autoscaling:updatepolicy:rollingupdate 命名空间

您还可以使用 `aws:autoscaling:updatepolicy:rollingupdate` 命名空间中的选项配置不可变更新。以下示例配置文件为配置更改启用不可变更新。

Example .ebextensions/immutable-updates.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Immutable
```

以下示例为配置更改和部署启用不可变更新。

Example .ebextensions/immutable-all.config

```
option_settings:
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Immutable
  aws:elasticbeanstalk:command:
    DeploymentPolicy: Immutable
```

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

更新 Elastic Beanstalk 环境的平台版本

Elastic Beanstalk 定期发布新平台版本以更新所有基于 Linux 和 Windows Server 的平台。新平台版本提供对现有软件组件的更新，并支持新功能和配置选项。要了解有关平台和平台版本的信息，请参阅Elastic Beanstalk 平台词汇表。

您可以使用 Elastic Beanstalk 控制台或 EB CLI 更新环境的平台版本。根据您要更新的平台版本，Elastic Beanstalk 建议使用以下两种方法之一来执行平台更新。

- 方法 1 - 更新环境的平台版本。当您更新到平台分支中的最新平台版本（具有相同的运行时、Web 服务器、应用程序服务器和操作系统，但不更改主要平台版本）时，我们建议使用此方法。这是最常见的常规平台更新。
- 方法 2 - 执行蓝/绿部署。当您更新到其他平台分支中的平台版本（具有不同的运行时、Web 服务器、应用程序服务器或操作系统版本）或更新到其他主要平台版本时，我们建议使用此方法。当您想要利用新的运行时功能或最新的 Elastic Beanstalk 功能时，或者当您想要脱离弃用或停用的平台分支时，这是一种很好的方法。

  从传统平台版本迁移需要蓝/绿部署，因为这些平台版本与当前支持的版本不兼容。

  将 Linux 应用程序迁移到 Amazon Linux 2 需要蓝绿部署，因为 Amazon Linux 2 平台版本与以前的 Amazon Linux AMI 平台版本不兼容。

有关选择最佳平台更新方法的更多帮助，请展开适用于您的环境平台的部分。

Docker

使用方法 1 执行平台更新。

多容器 Docker

使用方法 1 执行平台更新。

预配置的 Docker

请考虑以下情况：

- 如果您要将应用程序迁移到另一个平台，例如从 Go 1.4 (Docker) 迁移到 Go 1.11 或从 Python 3.4 (Docker) 迁移到 Python 3.6，请使用方法 2。
- 如果您要将应用程序迁移到其他 Docker 容器版本，例如从 Glassfish 4.1 (Docker) 迁移到 Glassfish 5.0 (Docker)，请使用方法 2。
- 如果您要更新到容器版本或主要版本没有变化的最新平台版本，请使用方法 1。

Go

使用方法 1 执行平台更新。

Java SE

请考虑以下情况：

- 如果您要将应用程序迁移到其他 Java 运行时版本，例如从 Java 7 迁移到 Java 8，请使用方法 2。
- 如果您要更新到运行时版本没有变化的最新平台版本，请使用方法 1。

使用 Tomcat 的 Java

请考虑以下情况：

- 如果您要将应用程序迁移到其他 Java 运行时版本或 Tomcat 应用程序服务器版本，例如从 Java 7 with Tomcat 7 迁移到 Java 8 with Tomcat 8.5，请使用方法 2。
- 如果您要跨主要 Java with Tomcat 平台版本（v1.x.x、v2.x.x 和 v3.x.x）迁移应用程序，请使用方法 2。
- 如果您要更新到运行时版本、应用程序服务器版本或主要版本没有变化的最新平台版本，请使用方法 1。

使用 IIS 的 Windows Server 上的 .NET

请考虑以下情况：

- 如果您要将应用程序迁移到其他 Windows 操作系统版本，例如从 Windows Server 2008 R2 迁移到 Windows Server 2016，请使用方法 2。
- 如果您要跨主要 Windows Server 平台版本迁移应用程序，请参阅从 Windows Server 平台更早的主要版本迁移，并使用方法 2。

- 如果您的应用程序当前运行在 Windows Server 平台 V2.x.x 上，并且您想要更新到最新的平台版本，请使用方法 1。

> **Note**
>
> v2 之前的 Windows Server 平台版本未从语义上进行版本控制。您只能启动其中每个 Windows Server 主要平台版本的最新版本，在升级之后无法回退。

Node.js

使用方法 2 执行平台更新。

PHP

请考虑以下情况：

- 如果您要将应用程序迁移到其他 Java 运行时版本，例如从 PHP 5.6 迁移到 PHP 7.2，请使用方法 2。
- 如果您要跨主要 PHP 平台版本（v1.x.x 和 v2.x.x）迁移应用程序，请使用方法 2。
- 如果您要更新到运行时版本或主要版本没有变化的最新平台版本，请使用方法 1。

Python

请考虑以下情况：

- 如果您要将应用程序迁移到其他 Python 运行时版本，例如从 Python 2.7 迁移到 Python 3.6，请使用方法 2。
- 如果您要跨主要 Python 平台版本（v1.x.x 和 v2.x.x）迁移应用程序，请使用方法 2。
- 如果您要更新到运行时版本或主要版本没有变化的最新平台版本，请使用方法 1。

Ruby

请考虑以下情况：

- 如果您要将应用程序迁移到其他 Ruby 运行时版本或应用程序服务器版本，例如从 Ruby 2.3 with Puma 迁移到 Ruby 2.6 with Puma，请使用方法 2。

- 如果您要跨主要 Ruby 平台版本（v1.x.x 和 v2.x.x）迁移应用程序，请使用方法 2。
- 如果您要更新到运行时版本、应用程序服务器版本或主要版本没有变化的最新平台版本，请使用方法 1。

方法 1 - 更新环境的平台版本

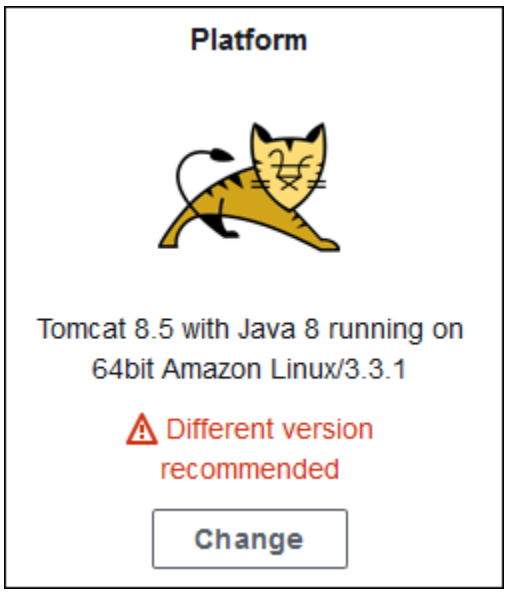
使用此方法可更新到环境的平台分支的最新版本。如果您之前已使用旧平台版本创建环境，或已从旧版本升级环境，则您也可以使用此方法恢复到先前的平台版本，前提是它在同一平台分支中。

更新环境的平台版本

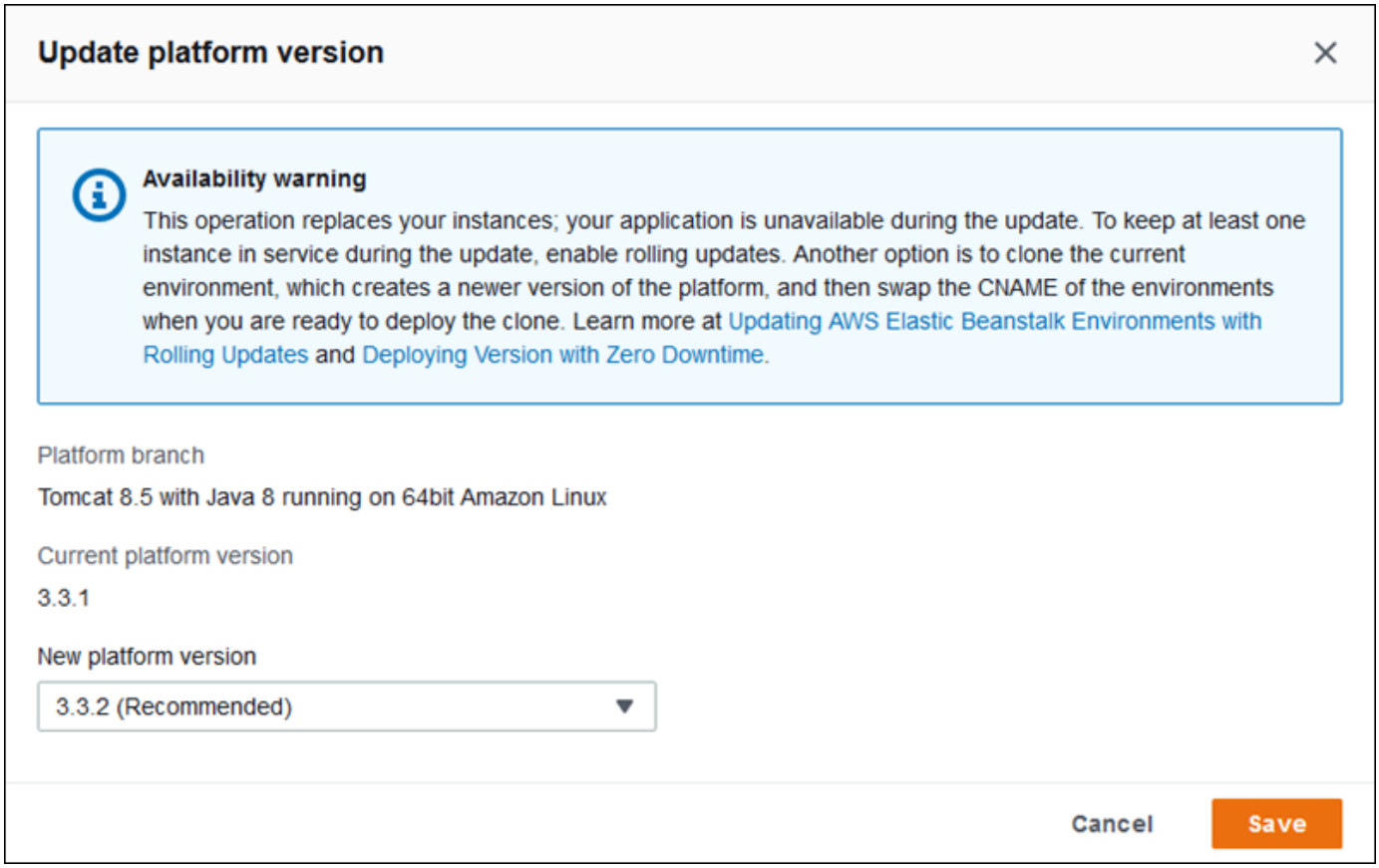
1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上的 Platform (平台) 下，选择 Change (更改)。



4. 在 Update platform version (更新平台版本) 对话框中，选择平台版本。将自动选择分支中的最新（推荐）平台版本。您可以更新到之前使用过的任何版本。



5. 选择 Save。

为了进一步简化平台更新，Elastic Beanstalk 可以为您管理它们。在可配置的每周维护时段内，您可以配置环境自动应用次版本和修补版本更新。Elastic Beanstalk 在应用托管更新时，不会导致停机时间或容量减少，并且当运行在新版本实例上的应用程序未能通过运行状况检查时，会立即取消更新。有关详细信息，请参阅托管平台更新。

方法 2 - 执行蓝/绿部署

使用此方法可更新到其他平台分支（具有不同的运行时、Web 服务器、应用程序服务器或操作系统版本）或更新到其他主要平台版本。当您想要利用新的运行时功能或最新的 Elastic Beanstalk 功能时，这通常是必需的。当您迁移出已弃用或已停用的平台分支时，也需要这样做。

当您跨主要平台版本或具有主要组件更新的平台版本进行迁移时，您的应用程序或其某些方面很可能无法在新平台版本上按预期运行，可能需要进行更改。

执行迁移前，请将本地开发计算机更新到较新的运行时版本以及您计划迁移到的平台的其他组件。验证应用程序是否仍按预期运行，并进行必要的代码修复和更改。然后使用以下最佳实践过程将环境安全地迁移到新平台版本。

将环境迁移到具有主要更新的平台版本

1. 创建新环境，使用新的目标平台版本，并将应用程序代码部署到其中。新环境应该在包含您要迁移的环境的 Elastic Beanstalk 应用程序中。此时不要终止现有环境。
2. 使用新环境迁移应用程序。具体而言：

   - 查找并修复在开发阶段未发现的任何应用程序兼容性问题。
   - 确保应用程序使用配置文件进行的任何自定义都能够在新环境中正常运行。这些可能包括选项设置、其他已安装的包、自定义安全策略以及环境实例上安装的脚本或配置文件。
   - 如果您的应用程序使用自定义 Amazon Machine Image (AMI)，请基于新平台版本的 AMI 创建新的自定义 AMI。要了解更多信息，请参阅“使用自定义 Amazon 系统映像 (AMI)”。具体而言，如果您的应用程序使用具有自定义 AMI 的 Windows Server 平台，并且您打算迁移到 Windows Server V2 平台版本，则需要执行此操作。在这种情况下，另请参阅从 Windows Server 平台更早的主要版本迁移。

   迭代测试并部署修复，直至对新环境上的应用程序满意。
3. 通过将新环境的 CNAME 与现有生产环境的 CNAME 交换，将新环境转换为生产环境。有关详细信息，请参阅使用 Elastic Beanstalk 进行蓝/绿部署。
4. 当您对处于生产模式的新环境状态感到满意之后，终止旧环境。有关详细信息，请参阅终止 Elastic Beanstalk 环境。

托管平台更新

AWS Elastic Beanstalk 定期发布平台更新以提供修复、软件更新和新功能。利用托管平台更新，您可以将环境配置为在计划的维护时段内自动升级到最新版本的平台。您的应用程序在更新过程中会继续提供服务，并且不会减少容量。托管更新对于单一实例和负载均衡环境均适用。

> **Note**
>
> 在早于版本 2 (v2) 的 Windows Server 平台版本上，此功能不可用。

您可以将环境配置为自动应用修补版本更新，或者同时应用修补版本更新和次版本更新。托管平台更新不支持跨平台分支的更新（对各种主要版本的平台组件（例如操作系统、运行时或 Elastic Beanstalk 组件）的更新），因为这些更新可能会引入无法向后兼容的更改。

您还可以将 Elastic Beanstalk 配置为在维护时段内替换环境中的所有实例，即使平台更新不可用也是如此。如果应用程序在长时间运行后遇到错误或内存问题，则替换环境中的所有实例会很有用。

在 2019 年 11 月 25 日或以后使用 Elastic Beanstalk 控制台创建的环境中，默认情况下，尽可能启用托管更新。托管更新需要启用增强型运行状况。默认情况下，当您选择某种配置预设时，将启用增强型运行状况，当您选择自定义配置时，将禁用增强型运行状况。控制台无法为不支持增强型运行状况的较旧平台版本启用托管更新，也无法在禁用增强型运行状况时启用托管更新。当控制台为新环境启用托管更新时，每周更新时段设置为一周中的任意一天的任意时间。更新级别设置为次要版本和补丁，并禁用实例替换。您可以在环境创建最终步骤之前禁用或重新配置托管更新。

对于现有环境，可以随时使用 Elastic Beanstalk 控制台配置托管平台更新。

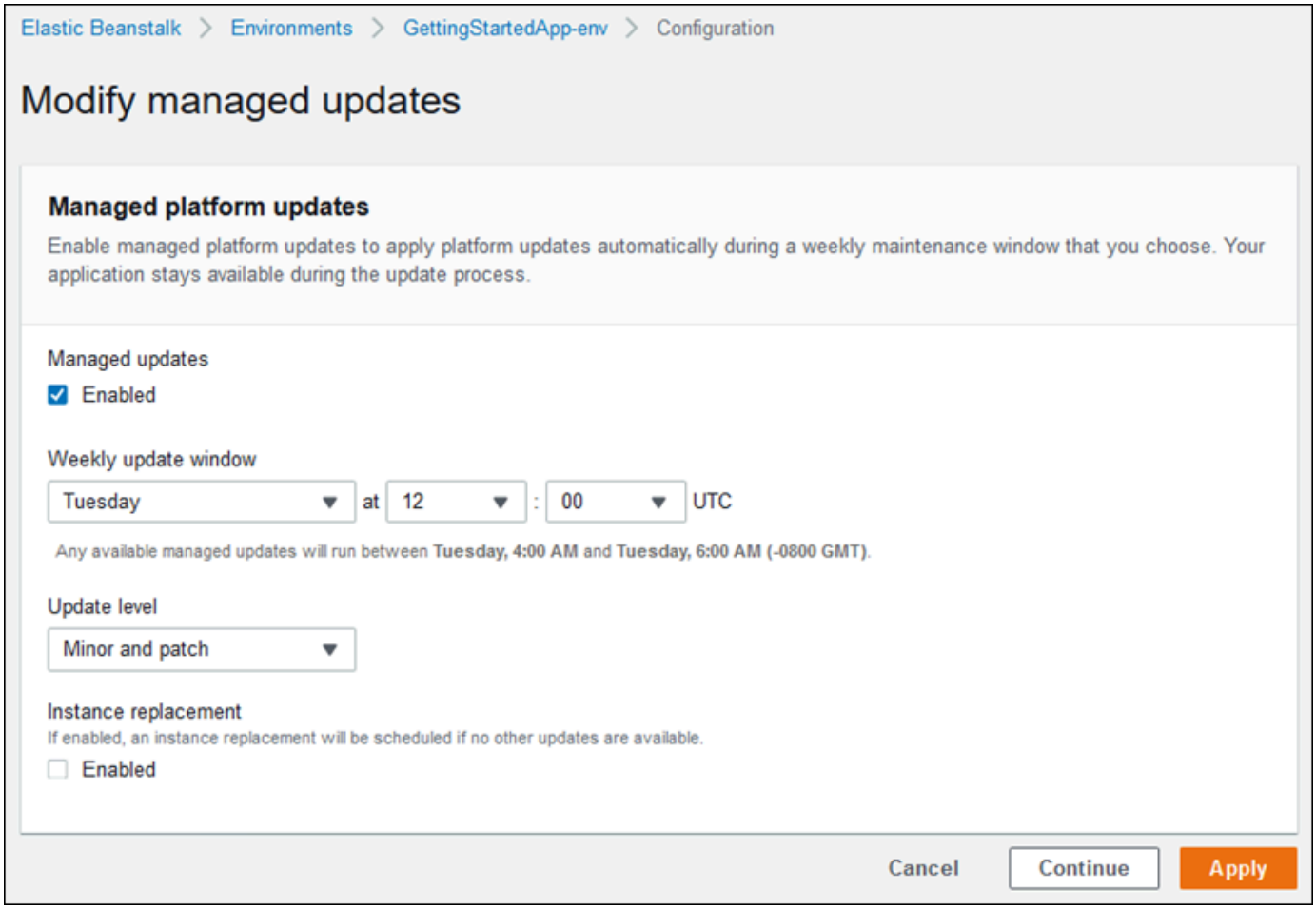
> **Important**
>
> 一个 AWS 账户中如果有大量 Beanstalk 环境，可能会在托管更新期间导致节流问题。大量是一个相对的量，取决于您为环境安排托管更新的紧密程度。一个账户中紧密安排的 200 多个环境可能会导致节流问题，但数量少也可能会出现问题。
> 为了平衡托管更新的资源负载，我们建议您将环境的定期维护窗口分散到一个账户中。
> 另外，可以考虑使用多账户策略。有关更多信息，请参阅AWS 白皮书和指南网站上的使用多个账户整理您的 AWS 环境。

配置托管平台更新

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Managed updates (托管更新) 类别中，选择 Edit (编辑)。
5. 禁用或启用托管更新。
6. 如果已启用托管更新，请选择维护时段，然后选择更新级别。
7. （可选）选择 Instance replacement (实例替换) 以启用每周实例替换。



8. 要保存更改，请选择页面底部的 Apply（应用）。

托管平台更新依赖于增强型运行状况报告来确定应用程序的运行状况足够良好并可认为平台更新成功。有关说明，请参阅启用 Elastic Beanstalk 增强型运行状况报告。

小节目录

执行托管平台更新所需的权限

Elastic Beanstalk 需要代表您启动平台更新的权限。为了获得这些权限，Elastic Beanstalk 会代入托管更新服务角色。当您为环境使用默认服务角色时，Elastic Beanstalk 控制台也会将其用作托管更新服务角色。控制台将 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 托管策略分配给您的服务角色。此策略具有 Elastic Beanstalk 执行托管平台更新所需的全部权限。

有关设置托管更新服务角色的其他方法的详细信息，请参阅the section called “服务角色”。

> **Note**
>
> 如果您使用配置文件扩展环境以包含其他资源，则可能需要向环境的托管更新服务角色添加权限。通常，当您在其他部分或文件中按名称引用这些资源时，您需要添加权限。

如果更新失败，您可以在托管更新页面上查找失败的原因。

托管更新维护时段

当 AWS 发布新版本的环境平台配置时，Elastic Beanstalk 会计划在下一个每周维护时段内执行托管平台更新。维护时段的长度为 2 小时。Elastic Beanstalk 在维护时段内开始执行计划的更新。此更新在该时段结束前可能无法完成。

> **Note**
>
> 在大多数情况下，Elastic Beanstalk 计划在即将到来的每周维护时段内进行托管更新。在计划托管更新时，系统会考虑更新安全和服务可用性的各个方面。在极少数情况下，更新可能不会安排在第一个即将到来的维护时段。如果发生这种情况，系统会在下一个维护时段内再次尝试。要手动应用托管更新，请选择 Apply now (立即应用)，如此页面的管理托管更新中所述。

次版本更新和修补版本更新

您可以启用托管平台更新以仅应用修补版本更新或同时应用次版本更新和修补版本更新。修补版本更新提供错误修复和性能改进，并且会包含对实例中的软件、脚本和配置选项的次要配置更改。次版本更新提供对新的 Elastic Beanstalk 功能的支持。您无法应用主版本更新，这可能会做出无法与托管平台更新向后兼容的更改。

在平台版本号中，第二个数字为次更新版本，第三个数字为修补版本。例如，在平台版本 2.0.7 中，次版本为 0，修补版本为 7。

不可变的环境更新

托管平台更新执行不可变的环境更新以将环境升级到新的平台版本。不可变更新在更新您的环境时，在确认运行新版本的实例通过运行状况检查之前，不会禁用任何实例或修改您的环境。

在不可变更新中，Elastic Beanstalk 将部署与当前使用新平台版本运行的实例一样多的实例。新实例开始与运行旧版本的实例一起接受请求。如果一组新实例通过所有运行状况检查，则 Elastic Beanstalk 会终止旧的实例组，并且仅保留具有新版本的实例。

托管平台更新始终执行不可变更新，即使您在维护时段之外应用它们也是如此。如果您从 Dashboard (控制面板) 更改平台版本，则 Elastic Beanstalk 会应用您为配置更新选择的更新策略。

> **Warning**
>
> 某些策略会在部署或更新期间替换所有实例。这会导致丢失所有累积的 Amazon EC2 突发余额。这发生在以下情况下：
>
> - 已启用实例替换的托管平台更新
> - 不可变更新
> - 已启用不可变更新或流量拆分的部署

管理托管更新

Elastic Beanstalk 控制台在 Managed updates overview (托管更新概述) 页面上显示有关托管更新的详细信息。

查看有关托管更新的信息（控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Managed updates (托管更新)。

托管更新概览部分提供了有关计划的和挂起的托管更新的信息。History (历史记录) 部分列出成功的更新和失败的尝试。

您可以选择立即应用计划的更新，而不是等待进入维护时段。

立即应用托管平台更新（控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Managed updates (托管更新)。
4. 选择 Apply now (立即应用)。
5. 验证更新详细信息，然后选择 Apply (应用)。

在维护时段之外应用托管平台更新时，Elastic Beanstalk 会执行不可变更新。如果您从控制面板更新环境平台或使用其他客户端更新环境平台，Elastic Beanstalk 将使用您为配置更改选择的更新类型。

如果您没有计划的托管更新，则您的环境可能已运行最新版本。导致未计划更新的其他原因包括：

- 次版本更新可用，但您的环境已配置为仅自动应用修补版本更新。
- 自发布更新后未扫描您的环境。Elastic Beanstalk 通常会每小时检查一次更新。
- 更新正在挂起或已在进行中。

在维护时段开始时或在您选择立即应用时，计划的更新在执行之前将进入挂起状态。

托管的操作选项命名空间

您可使用 `aws:elasticbeanstalk:managedactions` 和 `aws:elasticbeanstalk:managedactions:platformupdate` 命名空间中的配置选项启用和配置托管平台更新。

`ManagedActionsEnabled` 选项可启用托管平台更新。将此选项设置为 `true` 可启用托管平台更新，使用其他选项可配置更新行为。

使用 PreferredStartTime 可以用 *day*:*hour*:*minute* 格式配置每周维护时段的开始。

将 UpdateLevel 设置为 minor 或 patch 可同时应用次版本更新和修补版本更新，或仅应用修补版本更新。

在启用托管平台更新时，您可以通过将 InstanceRefreshEnabled 选项设置为 true 来启用实例替换。在启用此设置时，Elastic Beanstalk 每周在您的环境中运行一次不可变更新，而不管是否有新平台版本可用。

以下示例配置文件为维护时段在每个星期二 9:00 AM UTC 开始的修补版本更新启用托管平台更新。

Example .ebextensions/managed-platform-update.config

```
option_settings:
  aws:elasticbeanstalk:managedactions:
    ManagedActionsEnabled: true
    PreferredStartTime: "Tue:09:00"
  aws:elasticbeanstalk:managedactions:platformupdate:
    UpdateLevel: patch
    InstanceRefreshEnabled: true
```

从传统平台版本迁移应用程序

如果已部署使用传统平台版本的 Elastic Beanstalk 应用程序，您应当将应用程序迁移到使用非传统平台版本的新环境，以便使用新功能。如果您不确定是否正在使用传统平台版本运行您的应用程序，则您可以在 Elastic Beanstalk 控制台中进行核实。有关说明，请参阅检查您使用的是否属于传统平台版本。

传统平台版本缺失什么新功能？

旧版平台不支持以下功能：

- 配置文件，如使用配置文件 (.ebextensions) 进行高级环境自定义主题中所述
- ELB 运行状况检查，如基本运行状况报告主题中所述
- 实例配置文件，如管理 Elastic Beanstalk 实例配置文件主题中所述
- VPC，如将 Elastic Beanstalk 和 Amazon VPC 结合使用主题中所述
- 数据套餐，如将数据库添加到 Elastic Beanstalk 环境主题中所述
- 工作线程套餐，如工作线程环境主题中所述
- 单个实例环境，如环境类型主题中所述

- 标签，如在 Elastic Beanstalk 环境中标记资源主题中所述
- 滚动更新，如Elastic Beanstalk 滚动环境配置更新主题中所述

为什么某些平台版本标记为传统版本？

一些较旧的平台版本不支持最新的 Elastic Beanstalk 功能。这些版本在 Elastic Beanstalk 控制台中的环境概述页面上标记为 (legacy) ((传统))。

检查您使用的是否属于传统平台版本

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，查看 Platform (平台) 名称。

   如果您在平台名称旁边看到 (legacy) ((传统))字样，则应用程序使用的是传统平台版本。

迁移应用程序

1. 将应用程序部署到新环境。有关说明，请转到创建 Elastic Beanstalk 环境。
2. 如果您有 Amazon RDS 数据库实例，请更新数据库安全组，以便访问新环境的 EC2 安全组。有关如何使用 AWS 管理控制台查找 EC2 安全组名称的说明，请参阅安全组。有关配置 EC2 安全组的详细信息，请转到 Amazon Relational Database Service 用户指南 中的使用数据库安全组的“向 Amazon EC2 安全组授予网络访问权限”部分。
3. 交换环境 URL。有关说明，请转到使用 Elastic Beanstalk 进行蓝/绿部署。
4. 终止旧环境。有关说明，请转到终止 Elastic Beanstalk 环境。

> **Note**
>
> 如果您使用的是 AWS Identity and Access Management (IAM)，则需要更新策略以包括AWS CloudFormation和 Amazon RDS（如果适用）。有关更多信息，请参阅将 Elastic Beanstalk 与 AWS Identity and Access Management。

将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2

本节介绍了如何使用以下迁移路径之一迁移应用程序。

- 从 Amazon Linux 2 平台分支迁移到 Amazon Linux 2023 平台分支。
- 从 Amazon Linux AMI（AL1）平台分支迁移到 Amazon Linux 2023（推荐）或 Amazon Linux 2 平台分支。

主题

从 Amazon Linux 2 迁移到 Amazon Linux 2023

本主题提供了将您的应用程序从 Amazon Linux 2 平台分支迁移到 Amazon Linux 2023 平台分支的指南。

差异和兼容性

在 Elastic Beanstalk AL2 平台和 AL2023 平台之间

Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台之间具有高度的兼容性。尽管还有一些差异需要注意：

- 实例元数据服务版本 1（IMDSv1）– DisableIMDSv1 选项设置在 AL2023 平台上默认为 `true`。在 AL2 平台上默认为 `false`。
- pkg-repo 实例工具 – pkg-repo 工具不适用于在 AL2023 平台上运行的环境。但是，您可以手动将软件包和操作系统更新应用于 AL2023 实例。有关更多信息，请参阅 Amazon Linux 2023 用户指南中的管理软件包和操作系统更新。
- Apache HTTPd 配置 – AL2023 平台的 Apache `httpd.conf` 文件中的一些配置设置与 AL2 的配置设置不同：
  - 默认情况下，拒绝访问服务器的整个文件系统。这些设置在 Apache 网站安全提示页面上的默认保护服务器文件中进行了描述。

- 阻止用户覆盖您配置的安全功能。该配置拒绝访问所有目录中的 `.htaccess` 设置，专门启用的目录除外。此设置在 Apache 网站安全提示页面上的保护系统设置中进行了描述。Apache HTTP 服务器教程：.htaccess 文件页面指出，此设置可能有助于提高性能。
- 拒绝访问带有名称模式 `.ht*` 的文件。此设置阻止 Web 客户端查看 `.htaccess` 和 `.htpasswd` 文件。

您可以更改您的环境的上述任何配置设置。有关更多信息，请参阅扩展 Elastic Beanstalk Linux 平台。展开反向代理主题以查看配置 Apache HTTPD 部分。

在 Amazon Linux 操作系统之间

有关 Amazon Linux 2 和 Amazon Linux 2023 操作系统之间差异的详细信息，请参阅《Amazon Linux 2023 用户指南》中的比较 Amazon Linux 2 和 Amazon Linux 2023。

有关 Amazon Linux 2023 的详细信息，请参阅《Amazon Linux 2023 用户指南》中的什么是 Amazon Linux 2023？。

一般迁移流程

当您准备开始生产时，Elastic Beanstalk 需要蓝/绿部署才能执行升级。以下是建议使用蓝/绿部署程序进行迁移的一般最佳实践步骤。

准备对您的迁移进行测试

在部署应用程序并开始测试之前，请查看上一节 差异和兼容性 中的信息。另请参阅《Amazon Linux 2023 用户指南》中的比较 Amazon Linux 2 与 Amazon Linux 2023 一节中引用的参考。记下此内容中应用于或可能应用于您的应用程序和配置设置的特定信息。

高级迁移步骤

1. 创建基于 AL2023 平台分支的新环境。
2. 将您的应用程序部署到目标 AL2023 环境。

   在您通过测试和调整新环境进行迭代时，您的现有生产环境将保持活动状态且不受影响。
3. 在新环境中全面测试您的应用程序。
4. 当您的目标 AL2023 环境准备好投入生产时，您将交换两个环境的规范名称记录（CNAME），将流量重定向到新的 AL2023 环境。

更详细的迁移步骤和最佳实践

有关蓝/绿部署过程的更多详细信息，请参阅 使用 Elastic Beanstalk 进行蓝/绿部署。

有关更具体的指导和详细的最佳实践步骤，请参阅蓝/绿方法。

可帮助您规划迁移的更多参考

以下参考可以为规划迁移提供更多信息。

- AWS Elastic Beanstalk 平台中支持 Elastic Beanstalk 的平台
- 已停用平台分支历史记录
- the section called “Linux 平台”
- 平台停用常见问题

从 Amazon Linux AMI（AL1）迁移到 AL2 或 AL2023

如果您的 Elastic Beanstalk 应用程序基于 Amazon Linux AMI 平台分支，请使用此部分了解如何将应用程序的环境迁移到 Amazon Linux 2 或 Amazon Linux 2023。上一代平台分支基于 Amazon Linux AMI 而建，但现已停用。

强烈建议您迁移到 Amazon Linux 2023，因为它比 Amazon Linux 2 更新。Amazon Linux 2 操作系统将在 Amazon Linux 2023 之前终止支持，因此，如果您迁移到 Amazon Linux 2023，则将受益于更长的支持时间。

值得注意的是，Elastic Beanstalk Amazon Linux 2 和 Amazon Linux 2023 平台之间具有高度的兼容性。尽管有些方面确实存在差异：实例元数据服务版本 1（IMDSv1）选项默认支持 pkg-repo 实例工具的支持以及某些 Apache HTTPd 配置。有关更多信息，请参阅Amazon Linux 2023。

差异和兼容性

不能保证基于 AL2023/AL2 的平台分支向下兼容您的现有应用程序。另外，同样重要的要注意，即使您的应用程序代码成功部署到新平台版本，其行为和性能也可能会因操作系统和运行时而异。

虽然 Amazon Linux AMI 和 AL2023/AL2 使用的是相同的 Linux 内核，但它们在以下方面存在差异：初始化系统、`libc` 版本、编译器工具链和各种软件包。更多有关信息，请参阅 Amazon Linux 2 FAQs。

Elastic Beanstalk 服务还更新了特定于平台的运行时版本、构建工具和其他依赖项。

因此，我们建议您花些时间在开发环境中彻底测试您的应用程序，并进行任何必要的调整。

一般迁移流程

当您准备开始生产时，Elastic Beanstalk 需要蓝/绿部署才能执行升级。以下是建议使用蓝/绿部署程序进行迁移的一般最佳实践步骤。

准备对您的迁移进行测试

在部署应用程序并开始测试之前，请查看 所有 Linux 平台的注意事项 中的信息，本主题后面将介绍这些信息。此外，请在以下 平台特定注意事项 部分中查看适用于您的平台的信息。记下此内容中应用于或可能应用于您的应用程序和配置设置的特定信息。

高级迁移步骤

1. 创建基于 AL2 或 AL2023 平台分支的新环境。建议您迁移到 AL2023 平台分支。
2. 将您的应用程序部署到目标 AL2023/AL2 环境。

   在您通过测试和调整新环境进行迭代时，您的现有生产环境将保持活动状态且不受影响。
3. 在新环境中全面测试您的应用程序。
4. 当您的目标 AL2023/AL2 环境准备好投入生产时，您将交换两个环境的规范名称记录（CNAME），将流量重定向到新环境。

更详细的迁移步骤和最佳实践

有关蓝/绿部署过程的更多详细信息，请参阅 使用 Elastic Beanstalk 进行蓝/绿部署。

有关更具体的指导和详细的最佳实践步骤，请参阅蓝/绿方法。

可帮助您规划迁移的更多参考

以下参考可以为规划迁移提供更多信息。

- 比较 Amazon Linux 2 与 Amazon Linux 2023 Amazon Linux 2023 用户指南。
- 《Amazon Linux 2023 用户指南》中的什么是 Amazon Linux 2023？
- AWS Elastic Beanstalk 平台中支持 Elastic Beanstalk 的平台
- 已停用平台分支历史记录
- the section called “Linux 平台”
- 平台停用常见问题

所有 Linux 平台的注意事项

下表讨论了在规划将应用程序迁移到 AL2023/AL2 时应注意的事项。这些注意事项适用于任何 Elastic Beanstalk Linux 平台，而不管平台采用何种特定的编程语言或应用程序服务器。

领域	更改和信息
配置文件	在 AL2023/AL2 平台上，您可以像以前一样使用配置文件，并且所有部分的工作方式相同。但是，某些特定设置的工作方式可能与早期 Amazon Linux AMI 平台上的工作方式不同。例如： • 使用配置文件安装的某些软件包可能在 AL2023/AL2 上不可用，或者其名称可能已更改。 • 某些特定于平台的配置选项已从其平台特定的命名空间移动到与平台无关的不同命名空间。 • `.ebextensions/nginx` 目录中提供的代理配置文件应移动到 `.platform/nginx` 平台挂钩目录。有关详细信息，请在 the section called “扩展 Linux 平台” 中展开反向代理配置部分。 我们建议使用平台挂钩在环境实例上运行自定义代码。您仍可以在 `.ebextensions` 配置文件中使用命令和容器命令，但这并不简单。例如，在 YAML 文件中编写命令脚本可能非常繁琐且很难测试。 对于需要引用 AWS CloudFormation 资源的任何脚本，您仍需要使用 `.ebextensions` 配置文件。
平台挂钩	AL2 平台引入了一种新的方法来扩展您的环境平台，即将可执行文件添加到环境实例上的挂钩目录中。对于以前的 Linux 平台版本，您可能使用了自定义平台挂钩。这些挂钩不是为托管平台设计的，因此不受支持，但在某些情况下使用可能会有效。对于 AL2023/AL2 平台版本，自定义平台挂钩不起作用。应当将所有挂钩迁移到新的平台挂钩。有关详细信息，请在 the section called “扩展 Linux 平台” 中展开平台挂钩部分。
支持的代理服务器	AL2023/AL2 平台版本与在 Amazon Linux AMI 平台版本中支持的每个平台支持相同的反向代理服务器。所有 AL2023/AL2 平台版本都使用 nginx 作为其默认反向代理服务器，但 ECS 和 Docker 平台除外。Tomcat、Node.js、PHP 和 Python 平台也支持将 Apache HTTPD 作为替代方案。所有平台都以一致的方式启用代理服务器配置，

领域	更改和信息
	如本节所述。但是，配置代理服务器与其在 Amazon Linux AMI 上时略有不同。以下是所有平台的区别： • 默认为 nginx - 所有 AL2023/AL2 平台版本上的默认代理服务器都为 nginx。在 Tomcat、PHP 和 Python 的 Amazon Linux AMI 平台版本上，默认代理服务器是 Apache HTTPD。 • 一致的命名空间 - 所有 AL2023/AL2 平台版本都使用 `aws:elasticbeanstalk:environment:proxy` 命名空间来配置代理服务器。在 Amazon Linux AMI 平台版本上，这是针对每个平台的决定，而 Node.js 使用了不同的命名空间。 • 配置文件位置 - 应将代理配置文件放在所有 AL2023/AL2 平台版本上的 `.platform/nginx` 和 `.platform/httpd` 目录中。在 Amazon Linux AMI 平台版本上，这些位置分别为 `.ebextensions/nginx` 和 `.ebextensions/httpd` 。 有关特定于平台的代理配置更改，请参阅the section called “平台特定注意事项”。有关 AL2023/AL2 平台上的代理配置的信息，请展开反向代理配置部分于 the section called “扩展 Linux 平台” 中。
代理配置更改	除了特定于每个平台的代理配置更改外，还有一些代理配置更改统一适用于所有平台。为了准确配置您的环境，请务必参考两者。 • 所有平台 – 请展开 反向代理配置部分于 the section called “扩展 Linux 平台” 中。 • 特定于平台 – 请参阅 the section called “平台特定注意事项”。
实例配置文件	AL2023/AL2 平台需要配置实例配置文件。如果没有该配置文件，环境创建可能会暂时成功，但是当需要实例配置文件的操作开始失败时，该环境可能会在创建后很快出现错误。有关详细信息，请参阅 the section called “实例配置文件”。
增强型运行状况	AL2023/AL2 平台版本默认启用增强型运行状况。如果您未使用 Elastic Beanstalk 控制台创建环境，则该功能是一种更改。无论平台版本如何，默认情况下，该控制台尽可能启用增强型运行状况。有关详细信息，请参阅 the section called “增强型运行状况报告和监控”。
自定义 AMI	如果您的环境使用自定义 AMI，请根据 AL2023/AL2 为使用 Elastic Beanstalk AL2023/AL2 平台的新环境创建新 AMI。

领域	更改和信息
自定义平台	AL2023/AL2 平台版本的托管 AMI 不支持自定义平台。

平台特定注意事项

本节讨论特定 Elastic Beanstalk Linux 平台特有的迁移注意事项。

Docker

基于 Amazon Linux AMI (AL1) 的 Docker 平台分支系列包括三个平台分支。我们建议为每个平台分支制定一条不同的迁移路径。

AL1 平台分支	迁移到 AL2023/AL2 的路径
由运行于 Amazon Linux AMI (AL1) 上的 Amazon ECS 托管的多容器 Docker	基于 ECS 的 Docker AL2023/AL2 平台分支 基于 ECS 的 Docker AL2023/AL2 平台分支将为多容器 Docker AL1 平台分支上运行的环境提供直接迁移路径。 • 类似于先前的多容器 Docker AL1 分支，AL2023/AL2 平台分支使用 Amazon ECS，将多个 Docker 容器协调部署到 Elastic Beanstalk 环境中的 Amazon ECS 集群。 • AL2023/AL2 平台分支支持先前的多容器 Docker AL1 分支的所有功能。 • AL2023/AL2 平台分支也支持相同的 `Dockerrun.aws.json` v2 文件。 有关将在多容器 Docker Amazon Linux 平台分支上运行的应用程序迁移到在 AL2023/AL2 上运行的 Amazon ECS 平台分支的更多信息，请参阅 ???。
在 Amazon Linux AMI (AL1) 上运行的 Docker	在 AL2023/AL2 平台分支上运行的 Docker 我们建议您将在基于预配置 Docker（Glassfish 5.0）或在 Amazon Linux AMI（AL1）上运行的 Docker 的环境中运行的应用程序，迁移到基于在 Amazon Linux 2 上运行的 Docker 或在 AL2023 上运行的 Docker 平台分支的环境中。 如果您的环境基于预配置 Docker (Glassfish 5.0) 平台分支，请参阅 the section called “教程—— GlassFish 在 Docker 上：2023 年亚马逊 Linux 之路”。

<table>
<tr><th>AL1 平台分支</th><th>迁移到 AL2023/AL2 的路径</th></tr>
<tr><td>运行 Amazon Linux AMI (AL1) 的预配置 Docker (Glassfish 5.0)</td><td>下表列出了特定于平台分支在 AL2023/AL2 上运行的 Docker 的迁移信息。<table>
<tr><th>领域</th><th>更改和信息</th></tr>
<tr><td>存储</td><td>Elastic Beanstalk 将 Docker 配置为使用存储驱动程序来存储 Docker 映像和容器数据。在 Amazon Linux AMI 上，Elastic Beanstalk 使用设备映射器存储驱动程序。为了提高性能，Elastic Beanstalk 预置了额外的 Amazon EBS 卷。在 AL2023/AL2 Docker 平台版本上，Elastic Beanstalk 使用 OverlayFS 存储驱动程序，并在不再需要单独卷的情况下实现更好的性能。

对于 Amazon Linux AMI，如果您使用 <code>BlockDeviceMappings</code> 命名空间 <code>aws:autoscaling:launchconfiguration</code> 选项向 Docker 环境添加自定义存储卷，建议您同时添加 Elastic Beanstalk 预配置的 <code>/dev/xvdcz</code> Amazon EBS 卷。Elastic Beanstalk 不再预配置此卷，因此您应从配置文件中将其删除。有关详细信息，请参阅 the section called “Amazon Linux AMI（在 Amazon Linux 2 之前）上的 Docker 配置”。</td></tr>
<tr><td>私有存储库身份验证</td><td>当您提供 Docker 生成的身份验证文件以连接到私有存储库时，您不再需要将其转换为 Amazon Linux AMI Docker 平台版本需要的旧格式。AL2023/AL2 Docker 平台版本支持新格式。有关详细信息，请参阅 the section called “使用私有存储库中的映像”。</td></tr>
<tr><td>代理服务器</td><td>AL2023/AL2 Docker 平台版本不支持不在代理服务器后面运行的独立容器。在 Amazon Linux AMI Docker 平台版本上，这一点过去可以通过 <code>aws:elasticbeanstalk:environment:proxy</code> 命名空间中 <code>ProxyServer</code> 选项的 none 值来实现。</td></tr>
</table></td></tr>
</table>

Go

下表列出了 Go 平台中 AL2023/AL2 平台版本的迁移信息。

领域	更改和信息
端口传递	在 AL2023/AL2 平台上，Elastic Beanstalk 不会通过 PORT 环境变量向应用程序进程传递端口值。您可以通过亲自配置 PORT 环境属性来模拟进程的此类行为。但是，如果您具有多个进程，并且您依赖于 Elastic Beanstalk 将增量端口值（5000、5100、5200 等）传递给进程，则应当修改您的实现。有关详细信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

Amazon Corretto

下表列出了 Java SE 平台中的 Corretto 平台分支的迁移信息。

领域	更改和信息
Corretto 与 OpenJDK	为了实现标准版（Java SE）Java 平台，AL2023/AL2 平台分支使用 Amazon Corretto，它是开放式 Java 开发工具包（OpenJDK）的 AWS 分发版。早期 Elastic Beanstalk Java SE 平台分支使用 Amazon Linux AMI 随附的 OpenJDK 软件包。
构建工具	AL2023/AL2 平台具有较新版本的构建工具：`gradle`、`maven` 和 `ant`。
JAR 文件处理	在 AL2023/AL2 平台上，如果源包（ZIP 文件）包含单个 JAR 文件而不包含其他文件，Elastic Beanstalk 不再将 JAR 文件重命名为 `application.jar` 。仅当提交 JAR 文件自身（而不是包含在 ZIP 文件中）时，才会重命名。
端口传递	在 AL2023/AL2 平台上，Elastic Beanstalk 不会通过 PORT 环境变量向应用程序进程传递端口值。您可以通过亲自配置 PORT 环境属性来模拟进程的此类行为。但是，如果您具有多个进程，并且您依赖于 Elastic Beanstalk 将增量端口值（5000、5100、5200 等）传递给进程，则应当修改您的实现。有关详细信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。
Java 7	Elastic Beanstalk 不支持 AL2023/AL2 Java 7 平台分支。如果您有 Java 7 应用程序，请将其迁移到 Corretto 8 或 Corretto 11。

Tomcat

下表列出了 Tomcat 平台中 AL2023/AL2 平台版本的迁移信息。

<table>
<tr><th>领域</th><th>更改和信息</th></tr>
<tr><td>配置选项</td><td>在 AL2023/AL2 平台版本上，Elastic Beanstalk 在 <code>aws:elasticbeanstalk:environment:proxy</code> 命名空间中仅支持配置选项和选项值的子集。以下是每个选项的迁移信息。

<table>
<tr><th>选项</th><th>迁移信息</th></tr>
<tr><td><code>GzipCompression</code></td><td>在 AL2023/AL2 平台版本上不支持。</td></tr>
<tr><td><code>ProxyServer</code></td><td>AL2023/AL2 Tomcat 平台版本同时支持 nginx 和 Apache HTTPD 版本 2.4 代理服务器。但是，不支持 Apache 版本 2.2。

在 Amazon Linux AMI 平台版本中，默认代理是 Apache 2.4。如果您使用了默认代理设置并添加了自定义代理配置文件，则代理配置在 AL2023/AL2 上仍然正常工作。但是，如果您使用了 <code>apache/2.2</code> 选项值，则现在必须将代理配置迁移到 Apache 版本 2.4。</td></tr>
</table>

AL2023/AL2 平台版本不支持 <code>aws:elasticbeanstalk:container:tomcat:jvmoptions</code> 命名空间中的 <code>XX:MaxPermSize</code> 选项。用于修改永久代大小的 JVM 设置仅适用于 Java 7 及更早版本，因此不适用于 AL2023/AL2 平台版本。</td></tr>
<tr><td>应用程序路径。</td><td>在 AL2023/AL2 平台上，您的环境的 Amazon EC2 实例上的应用程序目录路径为 <code>/var/app/current</code> 。在 Amazon Linux AMI 平台上，该路径为 <code>/var/lib/tomcat8/webapps</code> 。</td></tr>
</table>

Node.js

下表列出了 Node.js 平台中 AL2023/AL2 平台版本的迁移信息。

领域	更改和信息
已安装的 Node.js 版本	在 AL2023/AL2 平台上，Elastic Beanstalk 维护多个 Node.js 平台分支，并且仅在每个平台版本上安装与平台分支对应的最新版本的 Node.js 主版本。例如，Node.js 12 平台分支中的每个平台版本在默认情况下仅安装 Node.js 12.x.y。在 Amazon Linux

领域	更改和信息
	AMI 平台版本上，我们在每个平台版本上安装了多个 Node.js 版本的多个版本，并且只维护一个平台分支。 选择与您的应用程序所需的 Node.js 主版本对应的 Node.js 平台分支。
Apache HTTPD 日志文件名	在 AL2023/AL2 平台上，如果您使用 Apache HTTPD 代理服务器，则 HTTPD 日志文件名为 `access_log` 和 `error_log` ，这与支持 Apache HTTPD 的所有其他平台一致。在 Amazon Linux AMI 平台版本上，这些日志文件分别命名为 `access.log` 和 `error.log` 。 有关所有平台的日志文件名和位置的详细信息，请参阅 the section called “Elastic Beanstalk 如何设置 CloudWatch Logs”。

<table>
<tr><th>领域</th><th>更改和信息</th></tr>
<tr><td>配置选项</td><td>在 AL2023/AL2 平台上，Elastic Beanstalk 不支持 <code>aws:elasticbeanstalk:container:nodejs</code> 命名空间中的配置选项。其中一些选项具有备用项。以下是每个选项的迁移信息。
<table>
<tr><th>选项</th><th>迁移信息</th></tr>
<tr><td><code>NodeCommand</code></td><td>使用 <code>package.json</code> 文件中的 <code>Procfile</code> 或 <code>scripts</code> 关键字指定启动脚本。</td></tr>
<tr><td><code>NodeVersion</code></td><td>使用 <code>package.json</code> 文件中的 <code>engines</code> 关键字指定 Node.js 版本。请注意，您只能指定与平台分支对应的 Node.js 版本。例如，如果您使用的是 Node.js 12 平台分支，则只能指定 12.x.y Node.js 版本。有关详细信息，请参阅 the section called “使用 package.json 文件指定 Node.js 依赖项”。</td></tr>
<tr><td><code>GzipCompression</code></td><td>在 AL2023/AL2 平台版本上不支持。</td></tr>
<tr><td><code>ProxyServer</code></td><td>在 AL2023/AL2 Node.js 平台版本上，此选项移到 <code>aws:elasticbeanstalk:environment:proxy</code> 命名空间。您可以在 nginx（默认值）和 apache 之间进行选择。

AL2023/AL2 Node.js 平台版本不支持未在代理服务器后面运行的独立应用程序。在 Amazon Linux AMI Node.js 平台版本上，这一点过去可以通过 <code>aws:elasticbeanstalk:container:nodejs</code> 命名空间中 <code>ProxyServer</code> 选项的 none 值来实现。如果您的环境运行独立应用程序，请更新您的代码以侦听代理服务器（nginx 或 Apache）将流量转发到的端口。

<pre>var port = process.env.PORT || 5000;

app.listen(port, function() {
  console.log('Server running at http://127.0.0.1:%s',
 port);
});</pre></td></tr>
</table></td></tr>
</table>

PHP

下表列出了 PHP 平台中 AL2023/AL2 平台版本的迁移信息。

领域	更改和信息
PHP 文件处理	在 AL2023/AL2 平台上，将通过 PHP-FPM（CGI 进程管理器）处理 PHP 文件。在 Amazon Linux AMI 平台上，我们使用的是 mod_php（一个 Apache 模块）。
代理服务器	AL2023/AL2 PHP 平台版本同时支持 nginx 和 Apache HTTPD 代理服务器。默认为 nginx。 Amazon Linux AMI PHP 平台版本仅支持 Apache HTTPD。如果您添加了自定义 Apache 配置文件，则可以将 `aws:elasticbeanstalk:environment:proxy` 命名空间中的 `ProxyServer` 选项设置为 `apache`。

Python

下表列出了 Python 平台中 AL2023/AL2 平台版本的迁移信息。

领域	更改和信息
WSGI 服务器	在 AL2023/AL2 平台上，Gunicorn 是默认 WSGI 服务器。默认情况下，Gunicorn 侦听端口 8000。该端口可能与您的应用程序在 Amazon Linux AMI 平台上使用的端口不同。如果您正在设置 `aws:elasticbeanstalk:container:python` 命名空间的 `WSGIPath` 选项，请将该值替换为 Gunicorn 语法。有关详细信息，请参阅 the section called “Python 配置命名空间”。 此外，您可以使用 `Procfile` 指定和配置 WSGI 服务器。有关详细信息，请参阅 the section called “Procfile”。
应用程序路径。	在 AL2023/AL2 平台上，您的环境的 Amazon EC2 实例上的应用程序目录路径为 `/var/app/current`。在 Amazon Linux AMI 平台上，该路径为 `/opt/python/current/app`。
代理服务器	AL2023/AL2 Python 平台版本同时支持 nginx 和 Apache HTTPD 代理服务器。默认为 nginx。

领域	更改和信息
	Amazon Linux AMI Python 平台版本仅支持 Apache HTTPD。如果您添加了自定义 Apache 配置文件，则可以将 `aws:elasticbeanstalk:environment:proxy` 命名空间中的 `ProxyServer` 选项设置为 `apache`。

Ruby

下表列出了 Ruby 平台中 AL2023/AL2 平台版本的迁移信息。

领域	更改和信息
已安装 Ruby 版本	在 AL2023/AL2 平台上，Elastic Beanstalk 仅在每个平台版本上安装与平台分支对应的单个 Ruby 版本的最新版本。例如，Ruby 2.6 平台分支中的每个平台版本仅安装了 Ruby 2.6.x。在 Amazon Linux AMI 平台版本上，我们安装了多个 Ruby 版本的最新版本，例如 2.4.x、2.5.x 和 2.6.x。 如果您的应用程序使用的 Ruby 版本与您正在使用的平台分支不对应，建议您切换到具有适合您的应用程序的 Ruby 版本的平台分支。
应用程序服务器	在 AL2023/AL2 平台上，Elastic Beanstalk 仅在所有 Ruby 平台版本上安装 Puma 应用程序服务器。您可以使用 `Procfile` 启动其他应用程序服务器，并使用 `Gemfile` 来安装它。 在 Amazon Linux AMI 平台上，对于每个 Ruby 版本，我们支持两种平台分支 - 一种具有 Puma 应用程序服务器，另一种具有 Passenger 应用服务器。如果您的应用程序使用 Passenger，则可以将 Ruby 环境配置为安装和使用 Passenger。 有关更多信息以及示例，请参阅 the section called “Ruby 平台”。

平台停用常见问题

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 停用了所有基于 Amazon Linux AMI（AL1）的平台分支。

本常见问题中的回答参考了以下主题：

- Elastic Beanstalk 平台支持策略
- 已停用平台分支历史记录
- AWS Elastic Beanstalk 平台中支持 Elastic Beanstalk 的平台
- 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2
- Amazon Linux 2 常见问题。

1. 平台分支的停用意味着什么？

在公布的平台分支停用日期之后，除非您已经拥有基于该平台分支的活动环境，否则您将无法再基于已停用的平台分支创建新环境。有关更多信息，请参阅常见问题 11。Elastic Beanstalk 将停止为这些平台分支提供新的维护更新。建议不要在生产环境中使用已停用的平台分支。有关更多信息，请参阅常见问题 5。

2. 为什么 AWS 停用了基于 AL1 的平台分支？

当平台组件被其供应商弃用或停用时，Elastic Beanstalk 将停用平台分支。在本案例中，Amazon Linux AMI (AL1) 已于 2020 年 12 月 31 日终止标准支持。虽然 Elastic Beanstalk 在 2022 年仍将继续提供基于 AL1 的平台，但自上述日期以后，我们发布了具有最新功能的基于 AL2 和 AL2023 的平台。为了让客户继续受益于未来的最新安全性和功能，客户迁移到我们基于 AL2 或 AL2023 的平台至关重要。

3. 哪些平台分支已停用？

有关已停用的平台组件和平台分支的列表，请参阅 已停用平台分支历史记录。

4. 目前支持哪些平台？

请参阅 AWS Elastic Beanstalk 平台中支持 Elastic Beanstalk 的平台。

5. Elastic Beanstalk 是否会在停用后移除或终止我的环境中的任何组件？

我们针对已停用平台分支的政策不会删除对环境的访问权限，也不会删除资源。但是，由于供应商将其组件标记为寿命终止 (EOL)，Elastic Beanstalk 无法为已停用平台分支提供安全更新、技术支持或修补程序，因此基于已停用平台分支的环境最终可能会陷入无法预测的境地。例如，在已停用平台分支上运行的环境中，可能会出现有害且关键的安全漏洞。或者，如果环境随着时间推移变得与 Elastic Beanstalk 服务不兼容，则 EB API 操作可能会不再适用于该环境。基于已停用平台分支的环境保持活动状态的时间越长，出现这些类型风险的几率就越高。

如果您的应用程序在停用的平台分支上运行时遇到问题，并且您无法将其迁移到支持的平台，则需要考虑其他替代方案。解决方法包括将该应用程序封装到 Docker 映像中，以便将其以 Docker 容器的形式运行。这将允许客户使用我们的任何 Docker 解决方案，例如我们的 Elastic Beanstalk AL2023/AL2 Docker 平台，或者其他基于 Docker 的服务，例如亚马逊 ECS 或 Amazon EKS。非 Docker 替代方案包括我们的 AWS CodeDeploy 服务，它允许您完全自定义所需的运行时。

6. 我是否可以提交推迟停用日期的请求？

不可以。在停用日期后，现有环境将继续正常运行。只不过，Elastic Beanstalk 不会再提供平台维护和安全更新。因此，如果您仍在基于 AL1 的平台上运行应用程序，则迁移到 AL2 或 AL2023 是至关重要的。有关风险和解决方法的更多信息，请参阅常见问题 5。

7. 如果我无法及时完成 AL2 或 AL2023 迁移，有什么解决方法？

客户可以继续运行该环境，但我们强烈建议您制定计划将所有 Elastic Beanstalk 环境迁移到受支持的平台版本。这样做可将风险降至最低，并持续受益于更新的版本中提供的重要安全性、性能和功能增强。有关风险和解决方法的更多信息，请参阅常见问题 5。

8. 迁移到 AL2 或 AL2023 平台的建议流程是什么？

有关从 AL1 到 AL2023/AL2 的全面迁移说明，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。此主题解释了 Elastic Beanstalk 需要蓝/绿部署才能执行升级。

9. 如果我有在已停用平台上运行的环境，会有什么影响？

由于供应商将其组件标记为寿命终止 (EOL)，Elastic Beanstalk 无法为已停用平台分支提供安全更新、技术支持或修补程序，因此基于已停用平台分支的环境最终可能会陷入无法预测的境地。例如，在已停用平台分支上运行的环境中，可能会出现有害且关键的安全漏洞。或者，如果环境随着时间推移变得与 Elastic Beanstalk 服务不兼容，则 EB API 操作可能会不再适用于该环境。已停用平台分支上的环境保持活动状态的时间越长，出现这些类型风险的几率就越高。有关更多信息，请参阅常见问题 5。

10. 停用日期后 90 天会发生什么情况？

我们针对已停用平台分支的政策不会删除对环境的访问权限，也不会删除资源。但请注意，由于供应商将其组件标记为寿命终止 (EOL)，Elastic Beanstalk 无法为已停用平台分支提供安全更新、技术支持或修补程序，因此基于已停用平台分支的环境最终可能会陷入无法预测的境地。例如，在已停用平台分支上运行的环境中，可能会出现有害且关键的安全漏洞。或者，如果环境随着时间推移变得与 Elastic Beanstalk 服务不兼容，则 EB API 操作可能会不再适用于该环境。已停用平台分支上的环境保持活动状态的时间越长，出现这些类型风险的几率就越高。有关更多信息，请参阅常见问题 5。

11. 我是否可以基于已停用平台创建新环境？

如果您曾使用已停用平台分支使用同一账户并在同一区域创建了现有环境，则您可以基于该平台分支创建新环境。已停用的平台分支将无法在 Elastic Beanstalk 控制台中使用。但是，对于拥有基于已停用平台分支的现有环境的客户，它可通过 EB CLI、EB API 和 AWS CLI使用。此外，现有客户可以使用 Clone environment（克隆环境）和 Rebuild environment（重建环境）控制台。但请注意，基于已停用平台分支的环境最终可能会陷入无法预测的境地。有关更多信息，请参阅常见问题 5。

12. 如果我的现有环境在已停用的平台分支上运行，那么我何时才能基于已停用的平台分支创建新环境？ 我是否可以使用控制台、CLI 或 API 执行此操作？

您可以在退休日期之后创建环境。但请记住，已停用平台分支最终可能会陷入无法预测的境地。此类环境创建或处于活动状态的时间越长，该环境遇到意外问题的风险就越高。有关创建新环境的更多信息，请参阅常见问题 11。

13. 我是否可以克隆或重建基于已停用平台的环境？

可以。您可以使用 Clone environment（克隆环境）和 Rebuild environment（重建环境）控制台执行此操作。您还可以使用 EB CLI、EB API 和 AWS CLI。有关创建新环境的更多信息，请参阅常见问题 11。

但我们强烈建议您制定计划将所有 Elastic Beanstalk 环境迁移到受支持的平台版本。这样做可将风险降至最低，并持续受益于更新的版本中提供的重要安全性、性能和功能增强。有关风险和解决方法的更多信息，请参阅常见问题 5。

14. 在停用日期之后，基于已停用平台分支的 Elastic Beanstalk 环境的 AWS 资源会发生什么变化？ 例如，如果正在运行的 EC2 实例被终止，Elastic Beanstalk 能否启动基于 AL1 的新 EC2 实例来维持容量？

该环境的资源将保持活动状态，并继续正常运行。而且，是的，Elastic Beanstalk 将为该环境中的 AL1 EC2 实例自动扩展。但是，Elastic Beanstalk 将停止向该环境提供新的平台维护更新，这可能会导致该环境随着时间推移而陷入无法预测的境地。有关更多信息，请参阅常见问题 5。

15. AL2023/AL2 与 Amazon Linux AMI（AL1）操作系统之间的主要区别是什么？ Elastic Beanstalk AL2023/AL2 平台分支将受到什么影响？

虽然 Amazon Linux AMI 和 AL2023/AL2 使用的是相同的 Linux 内核，但它们在初始化系统、`libc` 版本、编译器工具链和各种软件包方面有所不同。更多有关信息，请参阅 Amazon Linux 2 FAQs。

Elastic Beanstalk 服务还更新了特定于平台的运行时版本、构建工具和其他依赖项。不能保证基于 AL2023/AL2 的平台分支向下兼容您的现有应用程序。此外，即使您的应用程序代码成功部署到新平台版本，其行为和性能也可能会因操作系统和运行时而异。有关您需要查看及测试的配置及自定义项的列表和描述，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

取消环境配置更新和应用程序部署

您可以取消正在进行的由环境配置更改触发的更新。您也可以取消正在进行的新应用程序版本的部署。例如，如果您决定要继续使用现有环境配置，而不是应用新环境配置设置，您可能想要取消更新。否则，您可能会发现您正在部署的新应用程序版本有导致其无法启动或正常运行的问题。通过取消环境更新或应用程序版本更新，您不必等到更新或部署过程完成就可以开始重新尝试更新环境或应用程序版本。

> **Note**
>
> 在清除阶段中删除不再需要的旧资源时，当最后一批实例已更新之后，不能再取消更新。

Elastic Beanstalk 将以执行最近一次成功更新时所用的同一方式执行回滚。例如，如果您在环境中启用了基于时间的滚动更新，Elastic Beanstalk 将在对一个实例批的回滚更改和对下一个实例批的回滚更改之间等待指定的暂停时间。或者，如果您最近开启了滚动更新，但上次您成功更新环境配置设置时没有进行滚动更新，Elastic Beanstalk 将同时对所有实例执行回滚。

一旦 Elastic Beanstalk 开始取消更新，您就无法阻止它回滚到之前的环境配置。回滚过程将继续，直至环境中的所有实例拥有上一个环境的配置或回滚过程失败。对于应用程序版本部署，取消部署只会停止部署；某些实例将拥有新应用程序版本，而其他实例将继续运行现有的应用程序版本。您稍后可以部署相同的应用程序版本或其他应用程序版本。

有关滚动更新的更多信息，请参阅Elastic Beanstalk 滚动环境配置更新。有关批处理应用程序版本部署的更多信息，请参阅部署策略和设置。

取消更新

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Actions（操作），然后选择 Abort current operation（中止当前操作）。

重建 Elastic Beanstalk 环境

如果您不使用 Elastic Beanstalk 功能修改或终止环境的底层AWS资源，您的 AWS Elastic Beanstalk 环境可能会变得不可用。如果出现这种情况，您可以重建环境，以尝试将其恢复为工作状态。重建环境会终止其所有资源，并使用具有相同配置的新资源替换它们。

您还可以在环境终止后的六周 (42 天) 内重建已终止的环境。重建时，Elastic Beanstalk 会尝试创建具有相同名称、ID 和配置的新环境。

重建运行环境

您可以通过 Elastic Beanstalk 控制台或使用 `RebuildEnvironment` API 重建环境。

重建运行环境 (控制台)

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 依次选择 Actions (操作)、Rebuild environment (重建环境)。
4. 选择 Rebuild (重建)。

重建运行环境会创建与旧资源具有相同配置的新资源；但是，资源 ID 是不同的，并且不会恢复旧资源上的任何数据。例如，重建具有 Amazon RDS 数据库实例的环境时会创建具有相同配置的新数据库，但不会向新数据库应用快照。

要使用 Elastic Beanstalk API 重建运行环境，请将 RebuildEnvironment 操作与 AWS CLI 或AWS 开发工具包配合使用。

```
$ aws elasticbeanstalk rebuild-environment --environment-id e-vdnftxubwq
```

重建已终止的环境

您可以使用 Elastic Beanstalk 控制台、EB CLI 或 RebuildEnvironment API 重建和恢复终止的环境。

> **Note**
>
> 除非您将自己的自定义域名用于已终止的环境，否则环境将使用 elasticbeanstalk.com 的子域。这些子域在 Elastic Beanstalk 区域内共享。因此，它们可以被同一区域的任何客户创建的任何环境使用。当您的环境被终止时，另一个环境可能会使用其子域。在这种情况下，重建将失败。
>
> 可以通过使用自定义域避免此问题。有关详细信息，请参阅 您的 Elastic Beanstalk 环境的域名。

最近终止的环境会在应用程序概览中显示一小时。在此期间，您可以在环境的控制面板中查看其事件，并使用 Restore environment (恢复环境) 操作重建它。

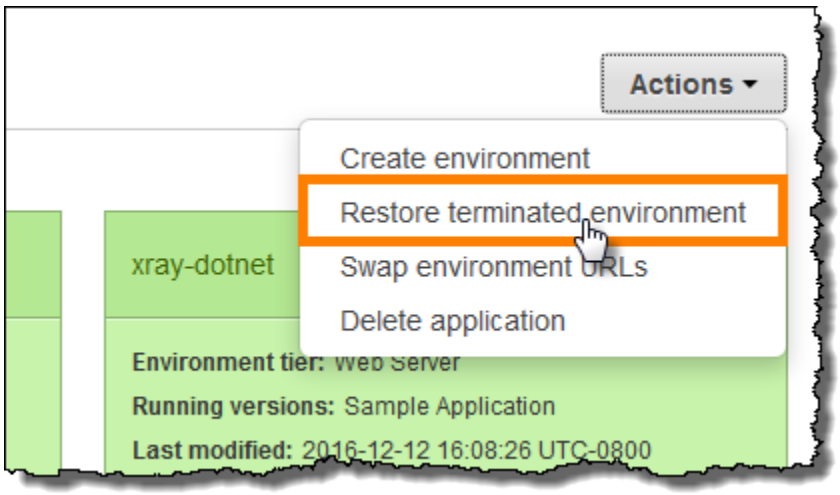
要重建不再可见的环境，请使用应用程序页面中的 Restore terminated environment (恢复终止的环境) 选项。

重建已终止的环境 (控制台)

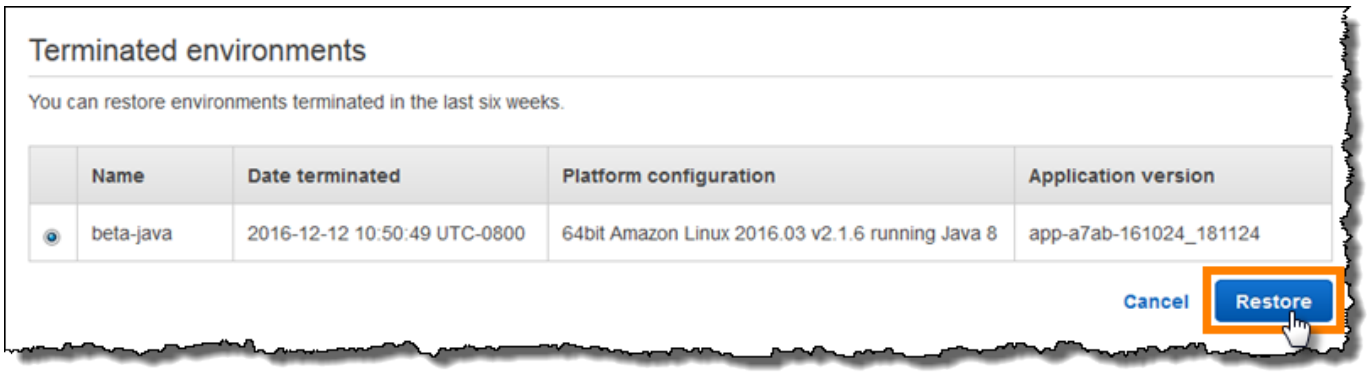
1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

   > **Note**
   >
   > 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 依次选择 Actions (操作)、Restore terminated environment (恢复终止的环境)。



4. 选择已终止的环境。
5. 选择 Restore（还原）。



Elastic Beanstalk 尝试创建具有相同名称、ID 和配置的新环境。在尝试重建时，如果存在具有相同名称或 URL 的环境，则重建将失败。删除已部署到环境的应用程序版本也会导致重建失败。

如果您使用 EB CLI 管理环境，请使用 eb restore 命令重建已终止的环境。

```
$ eb restore e-vdnftxubwq
```

参阅 eb restore 了解更多信息。

要使用 Elastic Beanstalk API 重建已终止的环境，请将 `RebuildEnvironment` 操作与 AWS CLI 或 AWS开发工具包配合使用。

```
$ aws elasticbeanstalk rebuild-environment --environment-id e-vdnftxubwq
```

环境类型

在 AWS Elastic Beanstalk 中，您可以创建负载均衡、可扩展的环境或单实例环境。您所需要的环境类型取决于您部署的应用程序。例如，您可以在单实例环境中开发和测试应用程序以节约成本，然后，当该应用程序可投入生产时，将该环境升级为负载均衡、可扩展的环境。

> **Note**
>
> 针对处理后台任务的 Web 应用程序的工作线程环境层不包括负载均衡器。不过，工作线程环境可通过将实例添加到 Auto Scaling 组进行有效扩展以便在负载需要它时处理来自 Amazon SQS 队列的数据。

负载平衡、可扩展的环境

负载均衡、可扩展的环境使用 Elastic Load Balancing 和 Amazon EC2 Auto Scaling 服务来预配置您所部署的应用程序需要的 Amazon EC2 实例。Amazon EC2 Auto Scaling 可自动启动其他实例，以适应应用程序上增大的负载。如果应用程序上的负载减小，Amazon EC2 Auto Scaling 将停止实例，但始终会保留您指定的最小运行实例数。如果您的应用程序需要通过在多个可用区运行的选项来实现可扩展性，请使用负载均衡、可扩展的环境。如果您不确定该选择哪种环境，可先选择一种，以后还可以根据需要切换环境类型。

单实例环境

单实例环境包括一个具有弹性 IP 地址的 Amazon EC2 实例。单实例环境没有负载均衡器，因此与负载均衡、可扩展的环境相比，有助于降低成本。尽管单实例环境会使用 Amazon EC2 Auto Scaling 服务，但最小实例数量、最大实例数量和所需容量全都设置为 1。这样设置的结果是，不会启动新实例来应对应用程序上增加的负载。

如果您预计生产应用程序的流量较小，或如果要进行远程开发，则可使用单实例环境。如果您不确定该选择哪种环境，您可以先选择一种，以后还可以根据需要切换环境类型。有关更多信息，请参阅更改环境类型。

更改环境类型

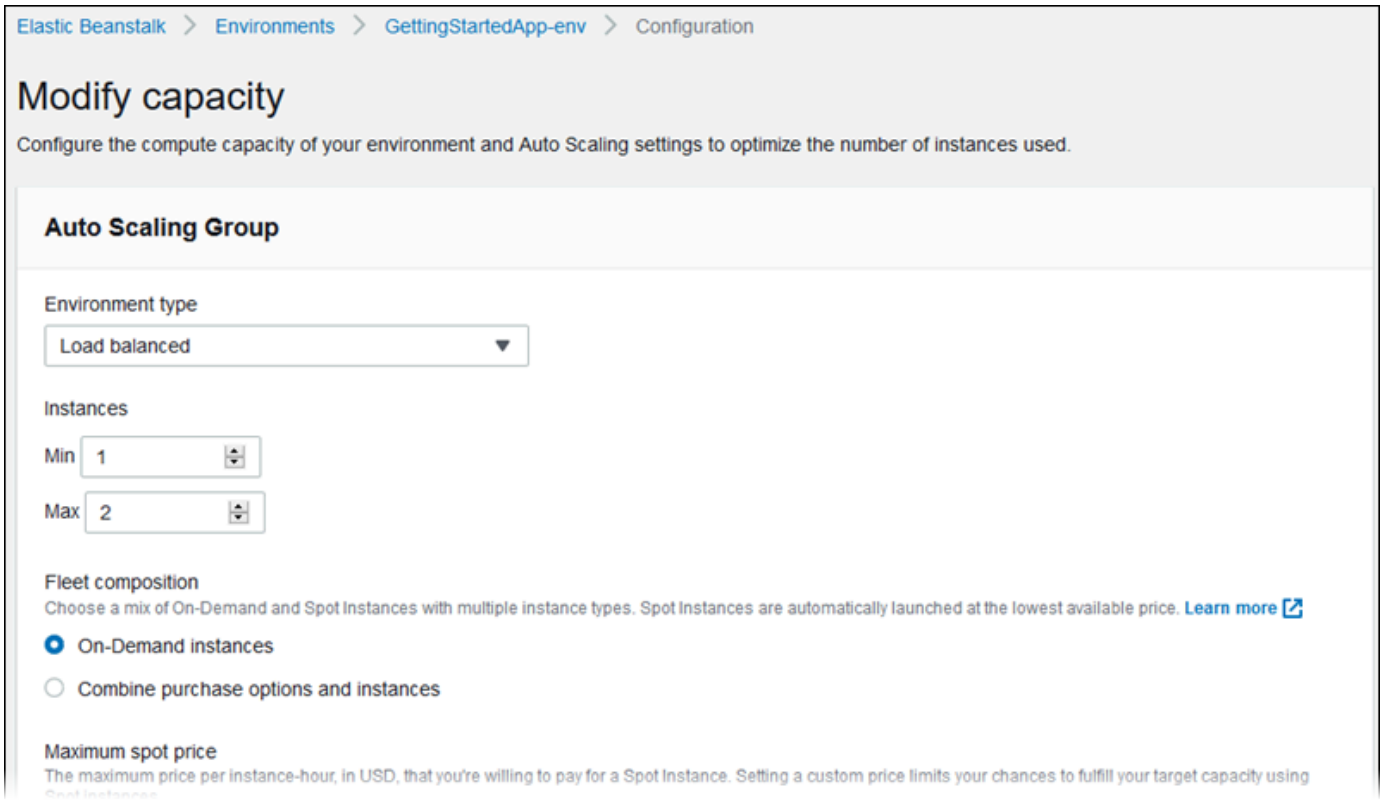
您可以通过编辑环境的配置，将环境类型更改为单实例或负载均衡、可扩展的环境。在一些情况下，您可能希望将环境类型从一种更改为另一种。例如，假如您为了节约成本，在单实例环境中完成了应用程序的开发和测试。当应用程序准备好投入生产时，您可以将环境类型更改为负载均衡、可扩展的环境，以便根据客户需求进行缩放。

更改环境类型

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Capacity (容量) 类别中，选择 Edit (编辑)。
5. 从 Environment Type (环境类型) 列表中，选择您需要的环境类型。



6. 选择 Save（保存）。

   请注意，在 Elastic Beanstalk 预置AWS资源期间，更新环境可能需要花费几分钟时间。

如果您的环境位于 VPC 中，请选择用于放置 Elastic Load Balancing 和 Amazon EC2 实例的子网。运行您的应用程序的所有可用区都必须包含这两者。有关详细信息，请参阅 将 Elastic Beanstalk 和 Amazon VPC 结合使用。

Elastic Beanstalk 工作线程环境

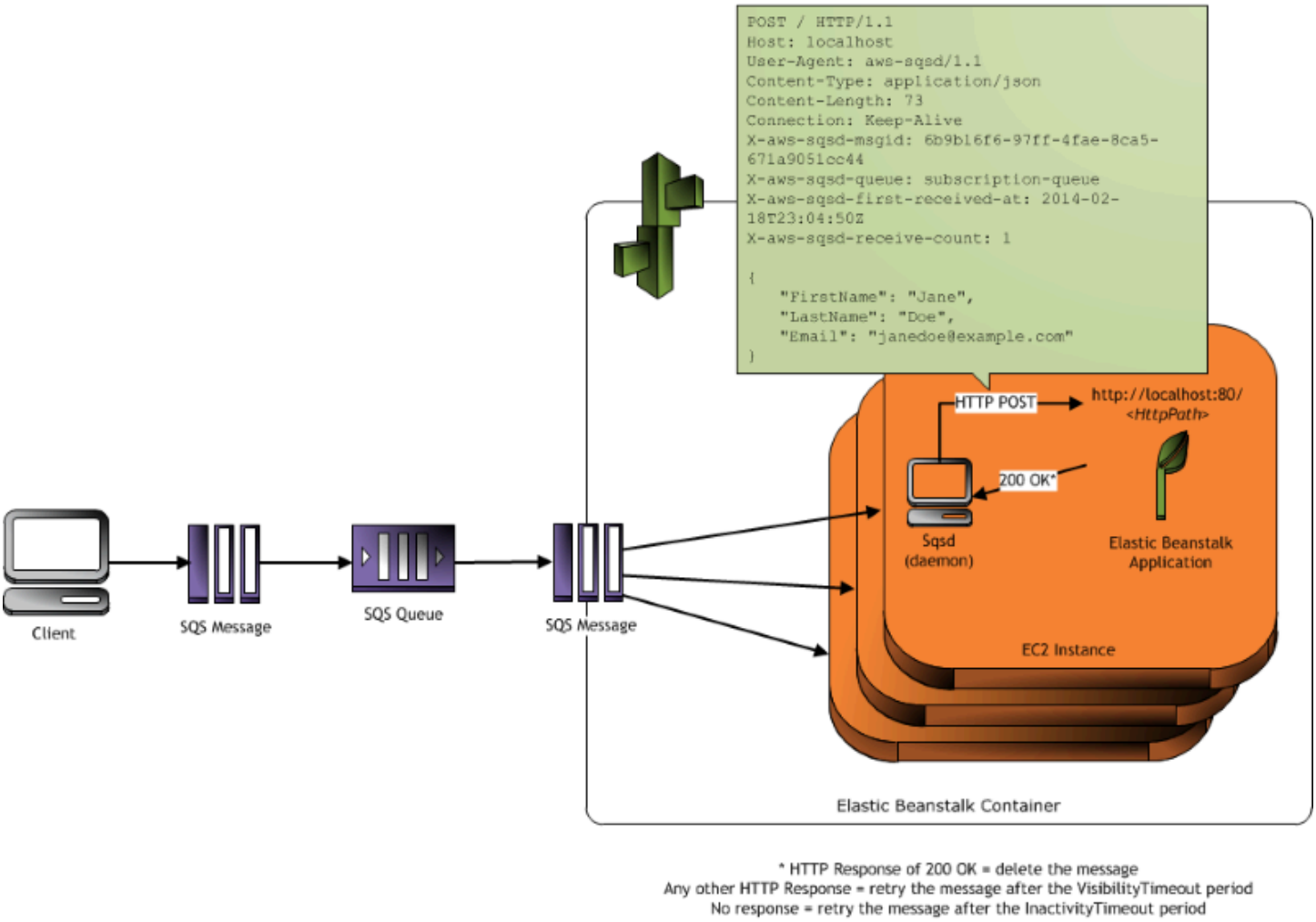
如果您的 AWS Elastic Beanstalk 应用程序执行的操作或工作流程需要很长时间才能完成，您可以将这些任务卸载到专用的工作线程环境。确保您的应用程序能够在负载下保持响应的常见方法是，将 Web 应用程序前端与执行阻止操作的过程分离开。

长时间运行的任务是指大大增加完成请求所需的时间的操作，例如，处理图像或视频、发送电子邮件或生成 ZIP 存档。这些操作可能只需花费 1 秒或 2 秒的时间即可完成，但对于要在 500 毫秒内完成的 Web 请求来说，几秒的延迟时间也是非常长的。

一种选择是在本地生成工作进程，返回成功值并异步处理任务。如果您的实例能够与其收到的所有任务保持同步，此选择便有效。不过，在高负载下，实例会填满后台任务并且不会响应优先级更高的请求。如果单个用户能够生成多个任务，负载增加可能不会对应用户增加，这会导致难以有效地向外扩展 Web 服务器。

要避免在本地运行长时间运行的任务，您可使用适用于编程语言的AWS开发工具包来将这些任务发送到 Amazon Simple Queue Service (Amazon SQS) 队列，并运行在一组单独的实例上执行这些任务的过程。然后，您采取适当的设计，让工作线程实例仅在能够运行项目时才从队列中提取项目，从而阻止项目将实例填满。

Elastic Beanstalk 工作线程环境通过管理 Amazon SQS 队列并在每个从队列中读取的实例上运行守护程序进程来简化此过程。当守护程序从队列中提取项目时，它会将 HTTP POST 请求本地发送到端口 80 上的 `http://localhost/` (正文中包含队列消息的内容)。您的应用程序只需执行长时间运行的任务来响应 POST 请求。可以配置守护程序以发布到其他路径，使用 application/JSON 之外的 MIME 类型，连接到现有队列或自定义连接（最大并发请求数）、超时和重试。



利用定期任务，您还可以将工作线程守护程序配置为根据 cron 计划对消息进行排队。每个定期任务均可向不同的路径发送 POST 请求。通过在定义每个任务的计划和路径的源代码中包含 YAML 文件来启用定期任务。

> **Note**
>
> Windows Server 平台上的 .NET 不支持工作线程环境。

小节目录

- 配置工作线程环境

工作线程环境 SQS 守护程序

工作线程环境运行 Elastic Beanstalk 提供的守护程序进程。此守护程序将定期更新以添加功能和修复 Bug。要获取最新版本的守护程序，请更新到最新的平台版本。

当工作线程环境中的应用程序返回 `200 OK` 响应以确认其已收到并成功处理了请求时，守护程序将向 Amazon SQS 队列发送 `DeleteMessage` 调用以便从队列中删除消息。如果应用程序返回 `200 OK` 之外的任何其他响应，则 Elastic Beanstalk 将会等待，然后在经过配置的 `ErrorVisibilityTimeout` 时间段后将消息放回到队列中。如果没有响应，则 Elastic Beanstalk 将会等待，在经过 `InactivityTimeout` 时间段后将消息放回到队列中，以便在处理期间可以对消息进行另一次尝试。

> **Note**
>
> Amazon SQS 队列的属性（消息顺序、至少一次传递和消息采样）可能会影响您为工作线程环境设计 Web 应用程序的方式。有关更多信息，请参阅《Amazon Simple Queue Service 开发人员指南》https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/Welcome.html中的分布式队列的属性。

Amazon SQS 会自动删除在队列中存在时间超过所配置的 `RetentionPeriod` 的消息。

守护程序将设置以下 HTTP 标头。

HTTP 标头

名称	值
`User-Agent`	`aws-sqsd` `aws-sqsd/1.1 1`
`X-Aws-Sqsd-Msgid`	SQS 消息 ID，用于检测消息风暴 (异常多的新消息).
`X-Aws-Sqsd-Queue`	SQS 队列的名称.

HTTP 标头	
`X-Aws-Sqsd-First-Received-At`	采用 ISO 8601 格式的 UTC 时间（当首次收到消息时）。
`X-Aws-Sqsd-Receive-Count`	SQS 消息接收计数.
`X-Aws-Sqsd-Attr-` *`message-attribute-name`*	分配给正在处理的消息的自定义消息属性。`message-attribute-name` 是实际消息属性名称。所有字符串和数字消息属性都将添加到标头。二进制属性已被放弃且未包含在标头中。
`Content-Type`	Mime 类型配置；默认情况下为 `application/json` 。

死信队列

Elastic Beanstalk 工作线程环境支持 Amazon Simple Queue Service (Amazon SQS) 死信队列。死信队列是其他 (源) 队列将因为某种原因而无法成功处理的消息发送到其中的队列。使用死信队列的主要好处是能够放弃和隔离未成功处理的消息。然后，您可以分析发送到死信队列的任何消息，以尝试确定它们未成功处理的原因。

如果您在创建工作线程环境层时指定了自动生成的 Amazon SQS 队列，则默认情况下会为工作线程环境启用死信队列。如果您为工作线程环境选择现有的 SQS 队列，则必须使用 SQS 单独配置死信队列。有关如何使用 SQS 配置死信队列的信息，请参阅使用 Amazon SQS 死信队列。

您不能禁用死信队列。无法交付的消息最终总会发送到死信队列。但是，您可以通过将 `MaxRetries` 选项设置为最大有效值 100 来有效禁用此功能。

如果没有为您的工作线程环境的 Amazon SQS 队列配置死信队列，则 Amazon SQS 会将消息保留在队列中，直至超过保留期。有关配置保留期的详细信息，请参阅the section called “配置工作线程环境”。

> **Note**
>
> Elastic Beanstalk `MaxRetries` 选项的功能与 SQS `MaxReceiveCount` 选项相同。如果您的工作线程环境不使用自动生成的 SQS 队列，请使用 SQS 中的 `MaxReceiveCount` 选项来有效禁用死信队列。有关更多信息，请参阅使用 Amazon SQS 死信队列。

有关 SQS 消息生命周期的更多信息，请转到消息生命周期。

定期任务

您可以在源包中名为 `cron.yaml` 的文件中定义定期任务，以便定期自动将作业添加到工作线程环境的队列中。

例如，以下 `cron.yaml` 文件创建两个定期任务。第一个任务每 12 小时运行一次，第二个任务在 UTC 时间每天晚上 11 点运行一次。

Example cron.yaml

```
version: 1
cron:
 - name: "backup-job"
   url: "/backup"
   schedule: "0 */12 * * *"
 - name: "audit"
   url: "/audit"
   schedule: "0 23 * * *"
```

对于每个任务，**name** 必须是唯一的。URL 是为触发作业而将 POST 请求发送到的路径。计划是一个用于确定任务运行时间的 CRON 表达式。

当任务运行时，守护程序会向环境的 SQS 队列发送一条消息，其标头指示需要执行的作业。环境中的任何实例均可选择消息和处理作业。

> **Note**
>
> 如果您使用现有 SQS 队列配置工作线程环境并选择 Amazon SQS FIFO 队列，则不支持定期任务。

Elastic Beanstalk 使用领导选择来确定工作线程环境中的哪个实例对定期任务进行排队。每个实例均尝试通过对 Amazon DynamoDB 表进行写入来变为领导。第一个获得成功的实例将成为领导，并且必须继续对表进行写入来维护领导状态。如果领导中断服务，则另一个实例将迅速成为领导。

对于定期任务，工作线程守护程序将另外设置以下领导。

HTTP 标头

名称	值
`X-Aws-Sqsd-Taskname`	对于定期任务，为要执行的任务的名称。
`X-Aws-Sqsd-Scheduled-At`	为定期任务安排的时间
`X-Aws-Sqsd-Sender-Id`	消息发送者的AWS账号

使用 Amazon CloudWatch 在工作线程环境层中自动扩展

Amazon EC2 Auto Scaling 和 CloudWatch 共同监控工作线程环境中运行的实例的 CPU 利用率。您为 CPU 容量配置的自动扩展限制将决定 Auto Scaling 组运行多少个实例来相应管理 Amazon SQS 队列中的消息吞吐量。每个 EC2 实例都会将其 CPU 利用率指标发布到 CloudWatch。Amazon EC2 Auto Scaling 从 CloudWatch 检索工作线程环境中所有实例的平均 CPU 利用率。您需要配置上限和下限阈值，以及要根据 CPU 容量添加或终止的实例的数量。当 Amazon EC2 Auto Scaling 检测到已达到 CPU 容量的指定上限阈值时，Elastic Beanstalk 会在工作线程环境中创建新实例。当 CPU 负载降到阈值之下时，将删除实例。

> **Note**
>
> 实例被终止时还未处理的消息将返回队列中，可由仍在运行的实例中的其他守护程序进行处理。

您还可以根据需要使用 Elastic Beanstalk 控制台、CLI 或选项文件设置其他 CloudWatch 警报。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon CloudWatch 结合使用和使用步进扩展策略创建 Auto Scaling 组。

配置工作线程环境

您可以编辑环境管理控制台中的 Configuration（配置）页面上的 Worker（工件）类别来管理工件环境的配置。

Elastic Beanstalk > Environments > GettingStartedApp-env > Configuration

Modify worker

You can create a new Amazon SQS queue for your worker application or pull work items from an existing queue. The worker daemon on the instances in your environment pulls an item from the queue and relays it in the body of a POST request to a local HTTP path relative to localhost.

Queue

Worker queue

Autogenerated queue

SQS queue from which to read work items.

Messages

HTTP path

/

The daemon pulls items from the Amazon SQS queue and posts them locally to this path.

MIME type

application/json

Change the MIME type of the POST requests that the worker daemon sends to your application.

HTTP connections

50

Maximum number of concurrent connections to the application.

Visibility timeout

300 seconds

The amount of time to lock an incoming message for processing before returning it to the queue.

Error visibility timeout

seconds

The amount of time to wait before resending a message after an error response from the application.

Advanced options

The following settings control advanced behavior of the worker tier daemon. Learn more

Max retries

10

Maximum number of retries after which the message is discarded.

Connection timeout

5

Inactivity timeout

> **Note**
>
> 您可以配置发布工作线程队列消息的 URL 路径，但不能配置 IP 端口。Elastic Beanstalk 始终在端口 80 上发布工作线程队列消息。工作线程环境应用程序或其代理必须侦听端口 80。

配置工作线程守护程序

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Worker (工作线程) 配置类别中，选择 Edit (编辑)。

Modify worker (修改工作线程) 配置页面具有以下选项。

在 Queue (队列) 部分中：

- 工作线程队列 – 指定守护程序从中读取的 Amazon SQS 队列。如果您有现有队列，可以选择现有队列。如果您选择 Autogenerated queue (自动生成的队列)，则 Elastic Beanstalk 将创建新的 Amazon SQS 队列和相应的 Worker queue URL (工作线程队列 URL)。

  > **Note**
  >
  > 选择 Autogenerated queue (自动生成的队列) 时，Elastic Beanstalk 创建的队列是标准 Amazon SQS 队列。选择现有队列时，可以提供标准或 FIFO Amazon SQS 队列。请注意，如果提供 FIFO 队列，不支持定期任务。

- Worker queue URL (工作线程队列 URL) – 如果您选择现有工作线程队列，则此设置将显示与该 Amazon SQS 队列关联的 URL。

在 Messages (消息) 部分中：

- HTTP path (HTTP 路径) – 指定将从 Amazon SQS 队列接收数据的应用程序的相对路径。该数据将插入到 HTTP POST 消息的消息正文中。默认值为 `/`。
- MIME type (MIME 类型) - 指示 HTTP POST 消息使用的 MIME 类型。默认值为 `application/json`。但是，因为您可以创建并指定您自己的 MIME 类型，所以任何值都是有效的。
- HTTP connections (HTTP 连接) – 指定守护程序可与 Amazon EC2 实例中的任何应用程序建立的最大并发连接数量。默认为 **`50`**。可以指定 **`1`** 到 **`100`**。
- Visibility timeout (可见性超时) – 指示锁定来自 Amazon SQS 队列的传入消息进行处理的时间（以秒为单位）。配置的时间量过后，会再次使消息在队列中可见，以供其他守护程序读取。选择一个值，该值大于应用程序处理消息所需的预期值，最多为 **`43200`** 秒。
- Error visibility timeout (错误可见性超时) – 指示在处理尝试由于显式错误而失败后，Elastic Beanstalk 将消息返回到 Amazon SQS 队列之前经过的时间（以秒为单位）。可以指定 **`0`** 到 **`43200`** 秒。

在 Advanced options (高级选项) 部分中：

- Max retries (最大重试次数) – 指定 Elastic Beanstalk 在将消息移到死信队列之前，尝试将消息发送到 Amazon SQS 队列的最大次数。默认值为 **`10`**。可以指定 **`1`** 到 **`100`**。

> **Note**
>
> Max retries（最大重试次数）选项仅适用于配置了死信队列的 Amazon SQS 队列。对于未配置死信队列的任何 Amazon SQS 队列，Amazon SQS 会将消息保留在队列中并持续处理，直至 Retention period（保留期）选项指定的期间过期为止。

- Connection timeout (连接超时) - 指定等待与应用程序成功建立连接的时间 (以秒为单位)。默认值为 **`5`**。可以指定 **`1`** 到 **`60`** 秒。
- Inactivity timeout (不活动超时) - 指示等待与应用程序的现有连接传回响应的时间 (以秒为单位)。默认值为 **`180`**。可以指定 **`1`** 到 **`36000`** 秒。
- Retention period (保留期) – 指示消息有效且正在进行处理的时间（以秒为单位）。默认值为 **`345600`**。可以指定 **`60`** 到 **`1209600`** 秒。

如果您使用现有的 Amazon SQS 队列，则您在创建工作线程环境时配置的设置可能会与直接在 Amazon SQS 中配置的设置冲突。例如，如果您在配置工作线程环境时所用的 `RetentionPeriod` 值高于在 Amazon SQS 中设置的 `MessageRetentionPeriod` 值，则 Amazon SQS 会在消息存在时间超过 `MessageRetentionPeriod` 时删除该消息。

反过来，如果您在工作线程环境设置中配置的 `RetentionPeriod` 值低于在 Amazon SQS 中设置的 `MessageRetentionPeriod` 值，则守护程序将先于 Amazon SQS 删除该消息。对于 `VisibilityTimeout`，您在工作线程环境设置中为守护程序配置的值将覆盖 Amazon SQS `VisibilityTimeout` 设置。请通过比较 Elastic Beanstalk 设置与 Amazon SQS 设置，确保合理地删除消息。

在 Elastic Beanstalk 环境之间创建链接

随着应用程序大小和复杂性的增加，您可能需要将其分割成具有不同开发和运营生命周期的组件。通过在明确定义的接口上运行多个彼此交互的较小服务，团队可以独立地工作并降低部署的风险。通过使用 AWS Elastic Beanstalk ，您可以链接环境以在彼此依赖的组件之间共享信息。

> **Note**
>
> Elastic Beanstalk 目前支持除多容器 Docker 以外的所有平台的环境链接。

借助环境链接，您可以将应用程序组件环境间的连接指定为命名引用。创建定义链接的环境时，Elastic Beanstalk 会设置与链接名称相同的环境变量。此变量的值是可用于连接其他组件 (可以是 Web 服务器或工作线程环境) 的终端节点。

例如，如果您的应用程序包含收集电子邮件地址的前端和向前端收集的电子邮件地址发送欢迎电子邮件的工作线程，则您可以在前端中创建一个指向工作线程的链接并让前端自动发现工作线程的终端节点 (队列 URL)。

在环境清单（一个名为 `env.yaml` 的 YAML 格式的文件，位于应用程序源的根目录中）中定义指向其他环境的链接。下面的清单定义了一个指向名为 worker 的环境的链接：

`~/workspace/my-app/frontend/env.yaml`

```
AWSConfigurationTemplateVersion: 1.1.0.0
EnvironmentLinks:
  "WORKERQUEUE": "worker"
```

当您使用包含上述环境清单的应用程序版本创建环境时，Elastic Beanstalk 会查找属于同一应用程序的名为 `worker` 的环境。如果该环境存在，Elastic Beanstalk 将创建一个名为 `WORKERQUEUE` 的环境属性。`WORKERQUEUE` 的值是 Amazon SQS 队列 URL。前端应用程序可以按照读取环境变量的方式读取此属性。有关详细信息，请参阅 环境清单 (env.yaml)。

要使用环境链接，请向您的应用程序源添加环境清单，然后使用 EB CLI AWS CLI 或 SDK 将其上传。如果您使用 AWS CLI 或 SDK，请在调用时设置process标志CreateApplicationVersion：

```
$ aws elasticbeanstalk create-application-version --process --application-name
 my-app --version-label frontend-v1 --source-bundle S3Bucket="DOC-EXAMPLE-
BUCKET",S3Key="front-v1.zip"
```

此选项告知 Elastic Beanstalk 在您创建此应用程序版本时验证您的源包中的环境清单和配置文件。如果您的项目目录中包含环境清单，EB CLI 会自动设置此标志。

使用任意客户端正常创建您的环境。当您需要终止环境时，请先终止带有链接的环境。如果某个环境链接到另一个环境，则 Elastic Beanstalk 会阻止被链接环境的终止操作。要覆盖这一保护设置，请使用 ForceTerminate 标志。此参数在 AWS CLI 中为 --force-terminate：

```
$ aws elasticbeanstalk terminate-environment --force-terminate --environment-name
 worker
```

配置 Elastic Beanstalk 环境

AWS Elastic Beanstalk 提供了多种选项，用于自定义环境中的资源，以及 Elastic Beanstalk 行为和平台设置。当您创建 Web 服务器环境时，Elastic Beanstalk 会创建多项资源来支持应用程序的运行。

- EC2 实例 - 配置为在您选择的平台上运行 Web 应用程序的 Amazon Elastic Compute Cloud (Amazon EC2) 虚拟机。

  各平台运行一组特定软件、配置文件和脚本以支持特定的语言版本、框架、Web 容器或其组合。大多数平台使用 Apache 或 NGINX 作为 Web 应用程序前的反向代理，向其转发请求、提供静态资产以及生成访问和错误日志。
- 实例安全组 - 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。通过此资源，HTTP 流量可以从负载均衡器到达运行您的 Web 应用程序的 EC2 实例。默认情况下，其他端口不允许流量进入。
- 负载均衡器 – 配置为向运行您的应用程序的实例分配请求的 Elastic Load Balancing 负载均衡器。负载均衡器还使您无需将实例直接公开在 Internet 上。
- 负载均衡器安全组 – 配置为允许端口 80 上的入站流量的 Amazon EC2 安全组。利用此资源，HTTP 流量可从 Internet 到达负载均衡器。默认情况下，其他端口不允许流量进入。
- Auto Scaling 组 – 配置为在实例终止或不可用时替换实例的 Auto Scaling 组。
- Amazon S3 存储桶 – 使用 Elastic Beanstalk 时创建的源代码、日志和其他构件的存储位置。
- Amazon CloudWatch CloudWatch 警报 — 两个警报，用于监控您环境中实例的负载，并在负载过高或过低时触发。警报触发后，您的 Auto Scaling 组会扩展或收缩以进行响应。
- AWS CloudFormation 堆栈 — Elastic AWS CloudFormation Beanstalk 用于在您的环境中启动资源并传播配置更改。这些资源在您可通过 AWS CloudFormation 控制台查看的模板中定义。
- 域名 – 一个域名，它以下面的形式路由到您的 Web 应用程序：*subdomain*.*region*.elasticbeanstalk.com。

> **Note**
>
> 为增强 Elastic Beanstalk 应用程序的安全性，已将 elasticbeanstalk.com 域注册到公共后缀列表（PSL）。为进一步增强安全性，如果您需要在 Elastic Beanstalk 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造（CSRF）攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

本主题重点介绍 Elastic Beanstalk 控制台中可用的资源配置选项。以下主题介绍了如何在控制台中配置您的环境。其中还介绍了与用于配置文件的控制台选项或 API 配置选项相对应的底层命名空间。要了解有关高级配置方法的信息，请参阅 配置环境（高级）。

主题

使用 Elastic Beanstalk 控制台进行环境配置

您可以使用 Elastic Beanstalk 控制台查看和修改您的环境及其资源的很多配置选项。您可以自定义环境在部署期间的行为，启用额外的功能，以及修改您在创建环境期间选择的实例类型和其他设置。

查看您的环境配置摘要

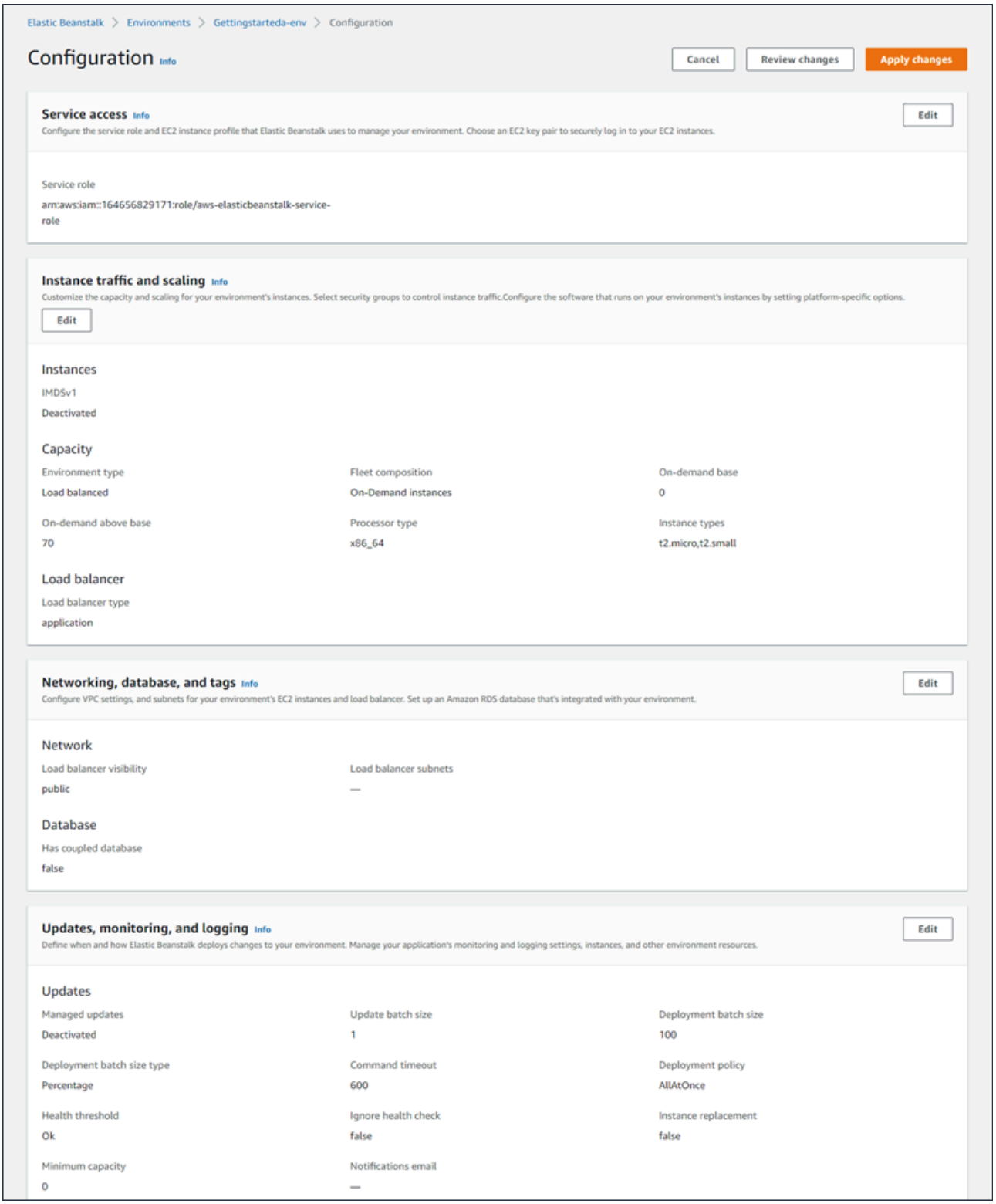
1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。

配置页面

配置概述页面显示一组配置类别。每个配置类别简要说明了一组相关的选项的当前状态。



在配置类别中选择 Edit (编辑) 以转到相关的配置页面，您可以在其中查看完整选项值并进行更改。查看并修改完选项后，您可以选择以下任一操作：

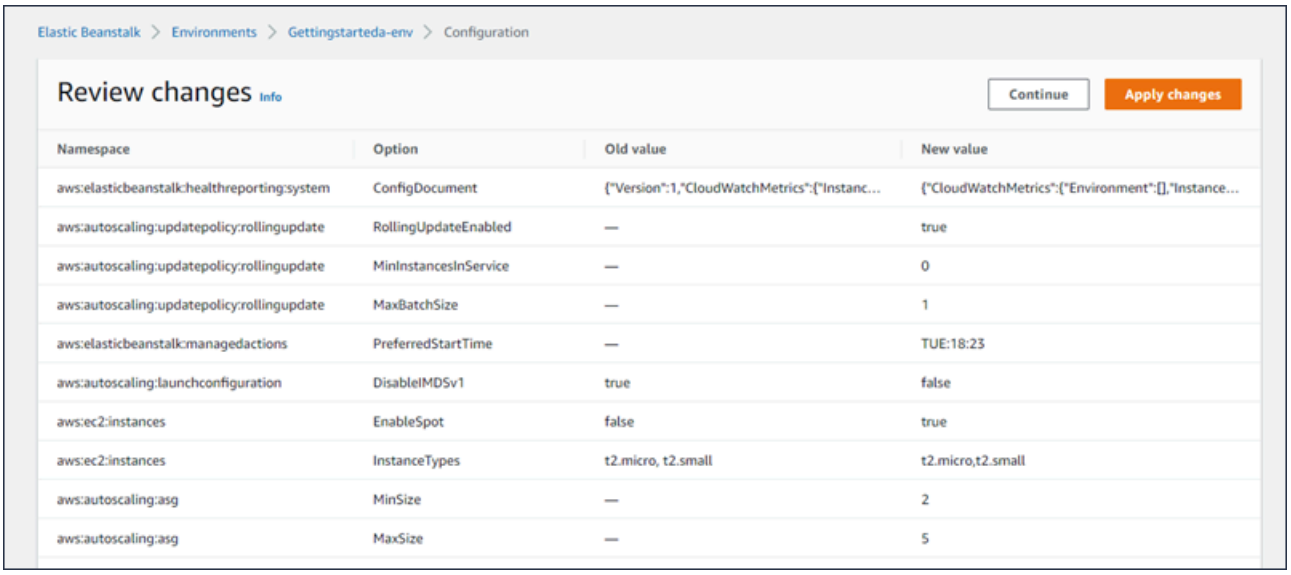
- 取消 – 返回环境的控制面板而不应用配置更改。在选择取消时，控制台将丢失您在任何配置类别中所做的任何待处理更改。

  您也可以通过选择其他控制台页面（如 Events（事件）或 Logs（日志））来取消配置更改。在这种情况下，如果有任何待处理的配置更改，控制台会提示您确认是否同意丢失这些更改。
- 检查更改 – 获取您在任何配置类别中所做的所有待处理更改的摘要。有关详细信息，请参阅 “Review changes (查看更改)”页面。
- Apply changes（应用更改）– 将您在任何配置类别中所做的更改应用于您的环境。在某些情况下，系统将提示您确认某个配置决策的结果。

“Review changes (查看更改)”页面

检查更改页面显示一个表，其中显示您在任何配置类别中所做的所有待处理选项更改（尚未应用于您的环境）。

表会以命名空间和选项的组合形式列出每个选项，Elastic Beanstalk 使用该组合来标识选项。有关详细信息，请参阅 配置选项。



查看完更改后，您可以选择以下任一操作：

- 继续 – 返回配置概述页面。然后，您可以继续进行更改或应用待处理更改。

- Apply changes（应用更改）– 将您在任何配置类别中所做的更改应用于您的环境。在某些情况下，系统将提示您确认某个配置决策的结果。

您的 Elastic Beanstalk 环境的 Amazon EC2 实例

创建 Web 服务器环境时， AWS Elastic Beanstalk 会创建一个或多个亚马逊弹性计算云 (Amazon EC2) 虚拟机，称为实例。

配置环境中的实例以便在您选择的平台上运行 Web 应用程序。您可以在创建环境时或在环境运行之后对环境实例的各种属性和行为进行更改。或者，您也可以通过修改部署到环境的源代码来进行这些更改。有关更多信息，请参阅 the section called “配置选项”。

> **Note**
>
> 您环境中的 Auto Scaling 组将管理运行应用程序的 Amazon EC2 实例。当您进行本页中所述的配置更改时，启动配置也将发生更改。启动配置是 Amazon EC2 启动模板或 Auto Scaling 组启动配置资源。此更改需要替换所有实例。它还会触发滚动更新或不可变更新，具体视配置而定。

Elastic Beanstalk 支持多种 Amazon EC2 实例购买选项：按需实例、预留实例 和 Spot 实例。按需实例是一种 pay-as-you-go资源，使用按需实例时无需长期承诺。预留实例是对环境中的匹配按需实例自动应用的预购账单折扣。Spot 实例是一种未使用的 Amazon EC2 实例，以低于按需价格提供。您可以通过设置单个选项在环境中启用 Spot 实例。您可以通过使用其他选项来配置 Spot 实例使用，包括混合使用按需实例和 Spot 实例。有关更多信息，请参阅Auto Scaling 组。

Sections

Amazon EC2 实例类型

当您创建新环境时，Elastic Beanstalk 会基于您选择的 Amazon EC2 实例类型预置 Amazon EC2 实例。您选择的实例类型决定了运行实例的主机硬件。EC2 实例类型可以按每种类型所基于的处理器架构进行分类。Elastic Beanstalk 支持基于以下处理器 AWS 架构的实例类型：Graviton 64 位 Arm 架构 (arm64)、64 位架构 (x86) 和 32 位架构 (i386)。创建新环境时，Elastic Beanstalk 会默认选择 x86 处理器架构。

> **Note**
>
> 大多数 Elastic Beanstalk 平台都不再支持 i386 32 位架构。我们建议您选择 x86 或 arm64 架构类型代替。Elastic Beanstalk 为 `aws:ec2:instances` 命名空间中的 i386 处理器实例类型提供配置选项。

指定 Elastic Beanstalk 环境配置中的所有实例类型必须具有相同类型的处理器架构。假设您将新实例类型添加到已经具有 t2.medium 实例类型（基于 x86 架构）的现有环境。您只能添加相同架构的其他实例类型，例如 t2.small。如果您想用来自不同架构的实例类型替换现有实例类型，则可以这样做。但是请确保命令中的所有实例类型都基于相同类型的架构。

在 Amazon EC2 推出新的兼容实例类型之后，Elastic Beanstalk 会定期增加对这些实例类型的支持。有关可用实例类型的信息，请参阅 Amazon EC2 用户指南中的实例类型或 Amazon EC2 用户指南中的实例类型。

> **Note**
>
> Elastic Beanstalk 现在在所有支持 Graviton 的地区的所有最新亚马逊 Linux 2 平台上都支持 Graviton。 AWS 有关使用基于 arm64 的实例类型创建 Elastic Beanstalk 环境的更多信息，请参阅 配置您环境的 Amazon EC2 实例。
> 创建在 arm64 架构上运行 Amazon EC2 实例的新环境，然后在 Elastic Beanstalk 中使用部署选项将现有的应用程序迁移到这些环境。
> 要了解有关基于 Graviton arm64 的处理器的更多信息，请参阅以下资源： AWS
>
> - 优点 — AWS Graviton 处理器
> - 入门和其他主题，例如特定语言的注意事项 — Grav it on 入门文章 AWS GitHub

配置您环境的 Amazon EC2 实例

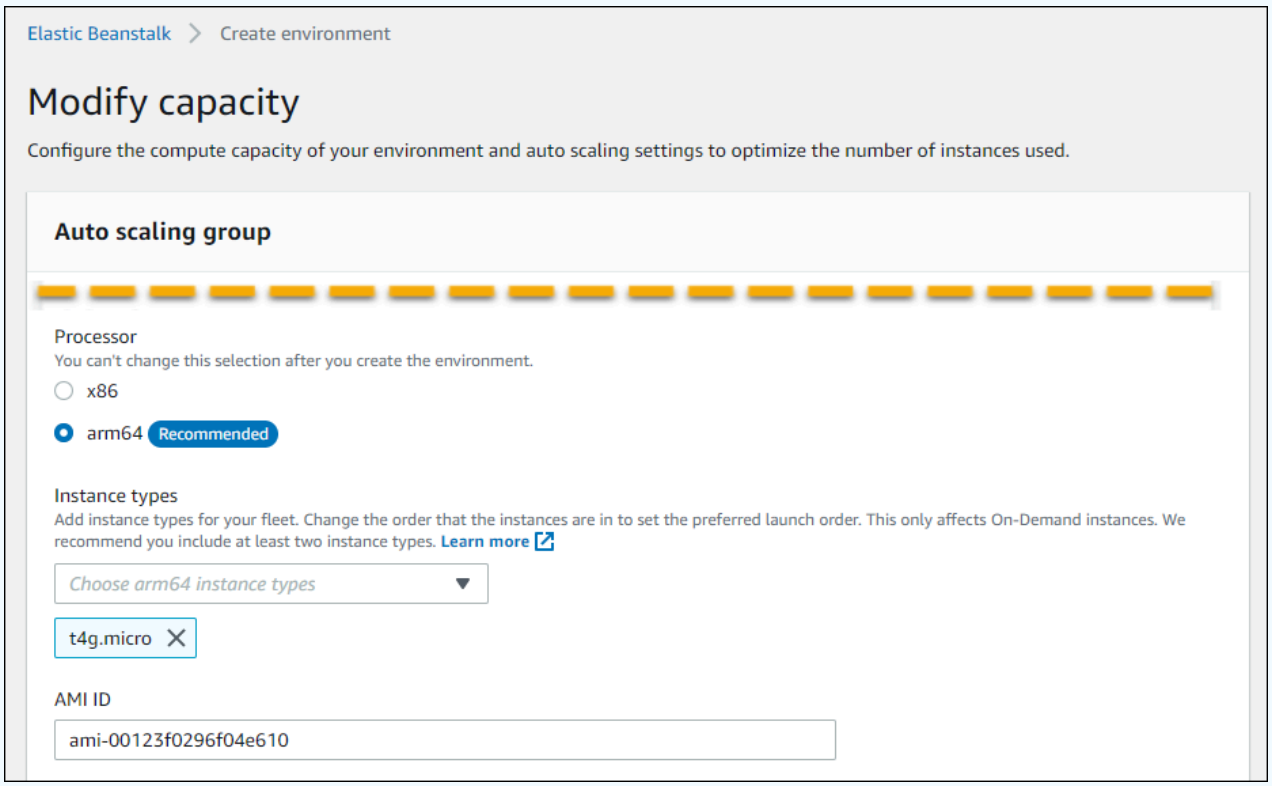
您可以在 Elastic Beanstalk 控制台中创建或修改 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

> **Note**
>
> 尽管 Elastic Beanstalk 控制台不提供更改现有环境处理器架构的选项，但你可以使用。 AWS CLI有关命令示例，请参见使用为您的环境配置 AWS EC2 实例 AWS CLI。

创建环境期间在 Elastic Beanstalk 控制台中配置 Amazon EC2 实例

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择环境。
3. 选择 Create a new environment (创建新环境) 以开始创建环境。
4. 在向导的主页上，在选择创建环境之前，选择配置更多选项。
5. 在 Instances (实例) 配置类别中，选择 Edit (编辑)。对此类别中的设置进行更改，然后选择应用。有关设置说明，请参阅本页上的the section called “实例类别设置”部分。
6. 在 Capacity（容量）配置类别中，选择 Edit（编辑）。对此类别中的设置进行更改，然后选择继续。有关设置说明，请参阅本页上的the section called “容量类别设置”部分。

> **选择处理器架构**
>
> 向下滚动到 Processor（处理器），为 EC2 实例选择处理器架构。控制台列出您之前在 Create environment（创建环境）面板中所选平台支持的处理器架构。
> 如果您没有看到所需的处理器架构，请返回配置类别列表选择支持该架构的平台。从 Modify Capacity（修改容量）面板，选择 Cancel（取消）。然后，选择 Change platform version（更改平台版本）选择新的平台设置。接下来，在 Capacity（容量）配置类别中选择 Edit（编辑）以再次查看处理器架构选择。



7. 选择保存，然后进行您的环境所需的任何其他配置更改。
8. 选择创建环境。

在 Elastic Beanstalk 控制台中配置运行环境的 Amazon EC2 实例

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

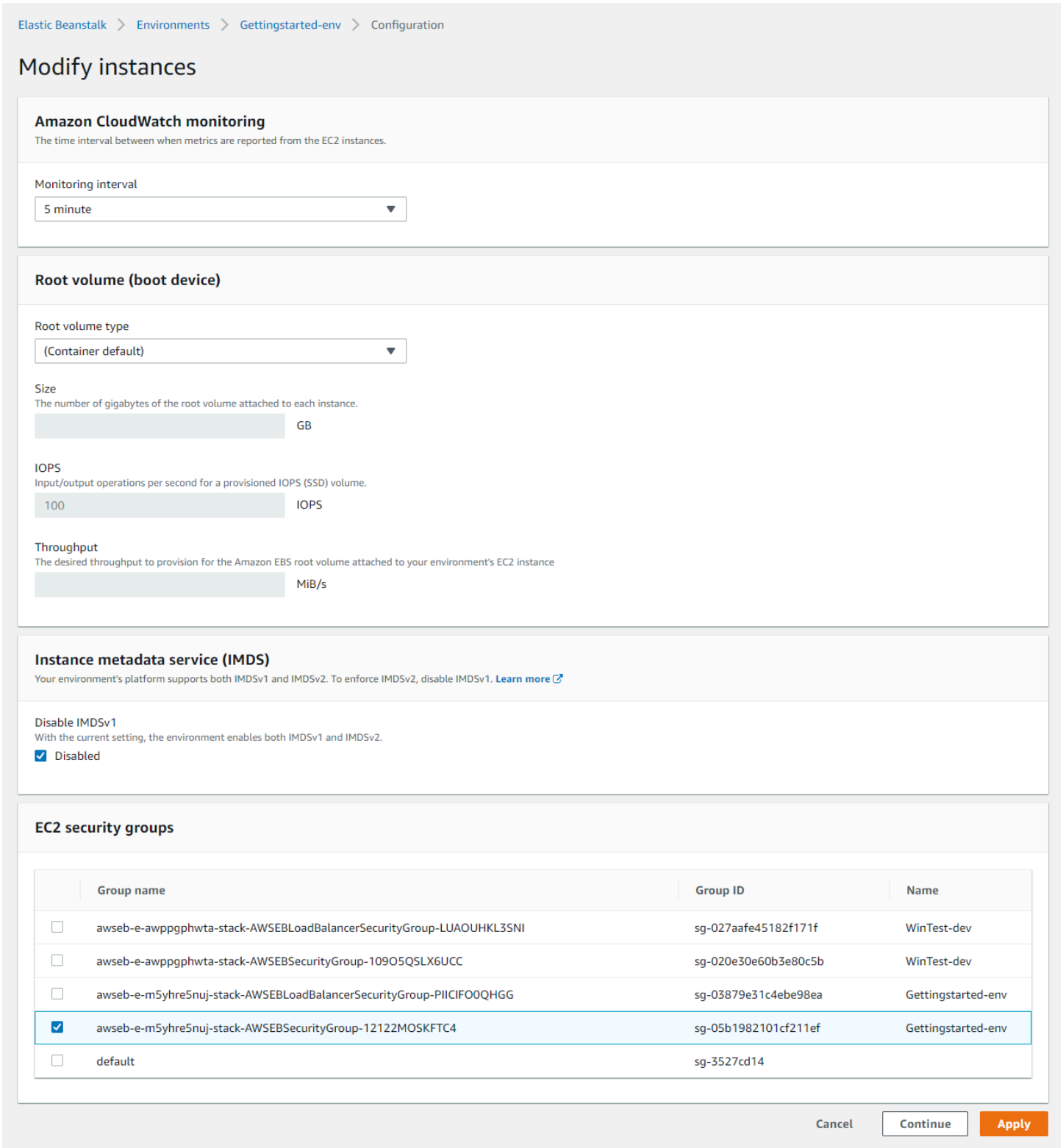
3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Instances (实例) 配置类别中，选择 Edit (编辑)。对此类别中的设置进行更改，然后选择应用。有关设置说明，请参阅本页上的the section called “实例类别设置”部分。
5. 在 Capacity（容量）配置类别中，选择 Edit（编辑）。对此类别中的设置进行更改，然后选择继续。有关设置说明，请参阅本页上的the section called “容量类别设置”部分。

实例类别设置

以下与 Amazon EC2 实例相关的设置在 Instances（实例）配置类别中提供。

选项

- 监控间隔
- 根卷（引导设备）
- 实例元数据服务
- 安全组


Elastic Beanstalk > Environments > Gettingstarted-env > Configuration
Modify instances
Amazon CloudWatch monitoring
The time interval between when metrics are reported from the EC2 instances.
Monitoring interval
5 minute
Root volume (boot device)
Root volume type
(Container default)
Size
The number of gigabytes of the root volume attached to each instance.
GB
IOPS
Input/output operations per second for a provisioned IOPS (SSD) volume.
100
IOPS
Throughput
The desired throughput to provision for the Amazon EBS root volume attached to your environment's EC2 instance
MiB/s
Instance metadata service (IMDS)
Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, disable IMDSv1. Learn more
Disable IMDSv1
With the current setting, the environment enables both IMDSv1 and IMDSv2.
Disabled
EC2 security groups
Group name
Group ID
Name
awseb-e-awppgphwta-stack-AWSEBLoadBalancerSecurityGroup-LUAOUHKL3SNI
sg-027aafe45182f171f
WinTest-dev
awseb-e-awppgphwta-stack-AWSEBSecurityGroup-109O5QSLX6UCC
sg-020e30e60b3e80c5b
WinTest-dev
awseb-e-m5yhre5nuj-stack-AWSEBLoadBalancerSecurityGroup-PIICIFO0QHGG
sg-03879e31c4ebe98ea
Gettingstarted-env
awseb-e-m5yhre5nuj-stack-AWSEBSecurityGroup-12122MOSKFTC4
sg-05b1982101cf211ef
Gettingstarted-env
default
sg-3527cd14
Cancel
Continue
Apply

监控间隔

默认情况下，您环境中的实例每隔五分钟向 Amazon CloudWatch 发布基本运行状况指标，无需支付额外费用。

要获得更详细的报告，您可以将监控间隔设置为 1 分钟，以提高环境中资源发布基本运行状况指标 CloudWatch 的频率。 CloudWatch 服务费适用于一分钟间隔指标。有关更多信息，请参阅 Amazon CloudWatch。

根卷（引导设备）

您的环境中的每个实例都配置了根卷。根卷是附加到实例的 Amazon EBS 块设备，用于存储操作系统、库、脚本以及您的应用程序源代码。默认情况下，所有平台为存储使用通用 SSD 块储存设备。

您可以修改根卷类型来使用机械硬盘存储或者预配置的 IOPS SSD 卷类型，并可根据需要增加卷大小。对于预配置 IOPS 卷，您还必须选择要预配置的 IOPS 数量。吞吐量仅适用于 gp3 固态硬盘卷类型。您可以输入要预置的所需吞吐量。它的范围可以在每秒 125 到 1000 兆字节 (MiB/s) 之间。选择能满足您的性能和价格要求的卷类型。

有关更多信息，请参阅《亚马逊 EC2 用户指南》中的 A mazon EBS 卷类型和亚马逊 EBS 产品详情。

实例元数据服务

实例元数据服务（IMDS）是实例上的组件，在实例上进行编码，用于安全访问实例元数据。代码可以使用两种方法之一，从正在运行的实例访问实例元数据。这两种方法是实例元数据服务版本 1 (IMDSv1) 或实例元数据服务版本 2 (IMDSv2)。IMDSv2 更加安全。禁用 IMDSv1 以强制实施 IMDSv2。有关更多信息，请参阅 the section called “IMDS”。

> **Note**
>
> 此配置页面上的 IMDS 部分仅为支持 IMDSv2 的平台版本显示。

安全组

附加到您实例的安全组决定允许哪些流量到达实例。这些实例还决定允许哪些流量离开实例。Elastic Beanstalk 会创建一个允许来自负载均衡器上标准端口 HTTP（80）和 HTTPS（443）流量的安全组。

您可以指定自己创建的其他安全组，允许来自其他端口或其他来源的流量。例如，您可以为 SSH 访问创建安全组，允许来自限定 IP 地址范围的端口 22 上的入站流量。或者，为了增强安全性，创建一个只允许您有权访问的堡垒主机的流量。

> **Note**
>
> 要允许环境 A 的实例与环境 B 的实例之间的流量，可向 Elastic Beanstalk 附加到环境 B 的安全组中添加一个规则。然后，您可以指定 Elastic Beanstalk 附加到环境 A 的安全组。这样就允许入站流量流入或出站流量流出环境 A 的实例。但是，这样做会在两个安全组之间建立依赖关系。如果之后尝试终止环境 A，Elastic Beanstalk 将无法删除该环境的安全组，因为环境 B 的安全组依赖于它。
>
> 因此，我们建议您首先创建单独的安全组。然后，将该安全组附加到环境 A，并在环境 B 的安全组规则中指定该组。

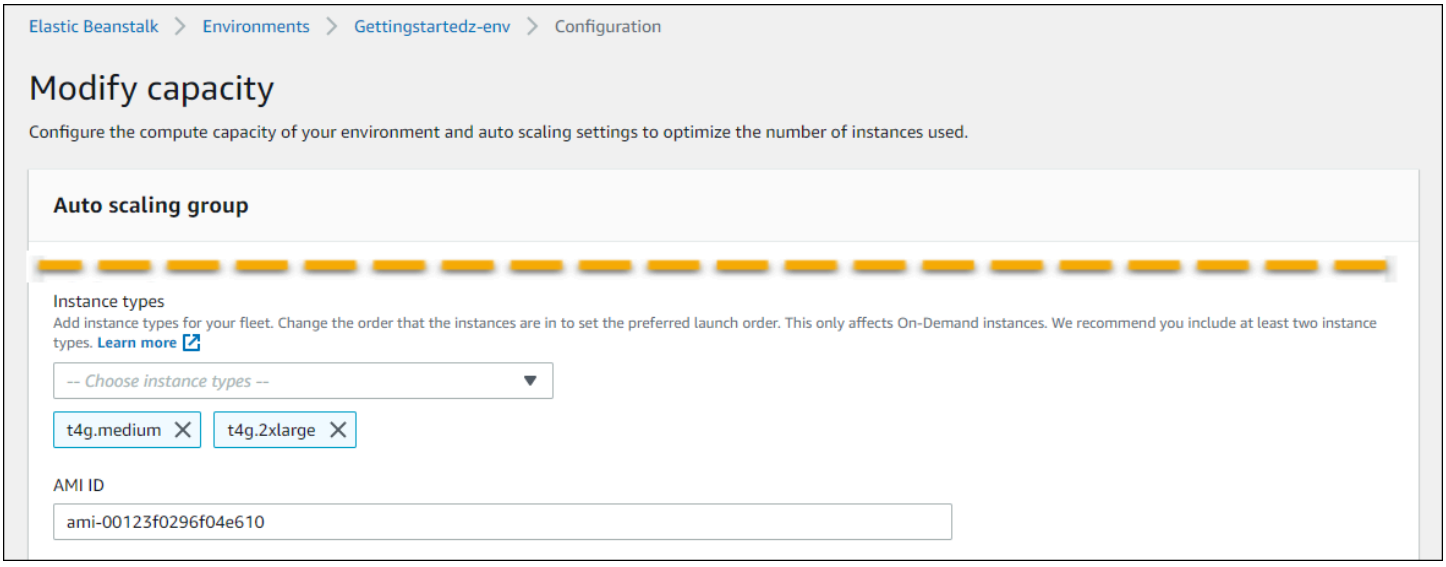
有关 Amazon EC2 安全组的更多信息，请参阅 Amazon EC2 用户指南中的亚马逊 EC2 安全组。

容量类别设置

以下与 Amazon EC2 实例相关的设置在 Capacity（容量）配置类别中提供。

Options

- 实例类型
- AMI ID



实例类型

Instance types（实例类型）设置确定启动用于运行您应用程序的 Amazon EC2 实例的类型。此配置页面显示实例类型的列表。您可以选择一种或多种实例类型。Elastic Beanstalk 控制台仅显示基于为您的环境配置的处理器架构的实例类型。因此，您只能添加相同处理器架构的实例类型，

> **Note**
>
> 尽管 Elastic Beanstalk 控制台不提供更改现有环境处理器架构的选项，但你可以使用。 AWS CLI有关命令示例，请参见使用为您的环境配置 AWS EC2 实例 AWS CLI。

选择充分配置的实例来运行具有高负载的应用程序，但配置不要过高以至于大部分时间空闲。对于开发用途，t2 系列的实例提供了中等处理能力，能够在短时间内突增。对于大规模、需要高可用性的应用程序，使用实例池来确保单个实例出现故障时，容量不会受到显著影响。首先使用您在正常时间的中等负载下用来运行五个实例的实例类型。如果任何实例失败，剩下实例可以接管剩余流量。通过容量缓冲区，当流量在高峰时间增长时，环境还可以扩展。

有关 Amazon EC2 实例系列和类型的更多信息，请参阅 Amazon EC2 用户指南中的实例类型或亚马逊 EC2 用户指南中的实例类型。要确定哪些实例类型符合您的要求及其支持的区域，请参阅 Amazon EC2 用户指南中的可用实例类型或 Amazon EC2 用户指南中的可用实例类型。

AMI ID

Amazon Machine Image（AMI）是 Elastic Beanstalk 用于在您的环境中启动 Amazon EC2 实例的 Amazon Linux 或 Windows Server 系统映像。Elastic Beanstalk 提供包含运行您的应用程序所需的工具和资源的系统映像。

Elastic Beanstalk 基于您选择的区域、平台版本和处理器架构选择适用于您的环境的默认 AMI。如果您创建了自定义 AMI，则使用自己的默认自定义 AMI ID 替换默认 AMI ID。

使用为您的环境配置 AWS EC2 实例 AWS CLI

使用 AWS 命令行界面 (AWS CLI) 使用命令行外壳中的命令创建和配置 Elastic Beanstalk 环境。本节提供 create-environment 和 update-environment 命令的示例。

前两个示例创建新环境。该命令指定基于 arm64 处理器架构的 Amazon EC2 实例类型 t4g.mall。Elastic Beanstalk 基于区域、平台版本和实例类型确定 EC2 实例的默认镜像 ID (AMI)。实例类型对应于处理器架构。`solution-stack-name` 参数适用于平台版本。

Example 1 — 创建一个新的基于 arm64 的环境（命名空间选项内联）

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role \
Namespace=aws:ec2:instances,OptionName=InstanceTypes,Value=t4g.small
```

作为替代方法，请使用 options.json 文件来指定命名空间选项，而不是将其内联。

Example 2 — 创建一个新的基于 arm64 的环境（**options.json** 文件中的命名空间选项）

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings file://options.json
```

Example

```
### example options.json ###
[
  {
    "Namespace": "aws:autoscaling:launchconfiguration",
    "OptionName": "IamInstanceProfile",
    "Value": "aws-elasticbeanstalk-ec2-role"
  },
  {
    "Namespace": "aws:ec2:instances",
    "OptionName": "InstanceTypes",
    "Value": "t4g.small"
  }
]
```

接下来的两个示例使用 update-environment 命令更新现有环境的配置。在本例中，我们添加了也基于 arm64 处理器架构的另一种实例类型。对于现有环境，添加的所有实例类型必须具有相同的处理器架构。如果您想用来自不同架构的实例类型替换现有实例类型，则可以这样做。但是请确保命令中的所有实例类型都具有相同的架构类型。

Example 3 — 更新现有的基于 arm64 的环境（命名空间选项内联）

```
aws elasticbeanstalk update-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role \
Namespace=aws:ec2:instances,OptionName=InstanceTypes,Value=t4g.small,t4g.micro
```

作为替代方法，请使用 options.json 文件来指定命名空间选项，而不是将其内联。

Example 4 — 更新现有的基于 arm64 的环境（**options.json** 文件中的命名空间选项）

```
aws elasticbeanstalk update-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings file://options.json
```

Example

```
### example options.json ###
[
  {
    "Namespace": "aws:autoscaling:launchconfiguration",
    "OptionName": "IamInstanceProfile",
    "Value": "aws-elasticbeanstalk-ec2-role"
  },
  {
    "Namespace": "aws:ec2:instances",
    "OptionName": "InstanceTypes",
    "Value": "t4g.small, t4g.micro"
```

```
  }
]
```

接下来的两个示例显示了更多 create-environment 命令。这些示例不提供 InstanceTypes 的值。未指定 InstanceTypes 值时，Elastic Beanstalk 默认使用基于 x86 的处理器架构。环境的 EC2 实例的镜像 ID (AMI) 将基于区域、平台版本和默认实例类型确定默认值。实例类型对应于处理器架构。

Example 5 — 创建一个新的基于 x86 的环境（命名空间选项内联）

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role
```

作为替代方法，请使用 options.json 文件来指定命名空间选项，而不是将其内联。

Example 6 — 创建一个新的基于 x86 的环境（**options.json** 文件中的命名空间选项）

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings file://options.json
```

Example

```
### example options.json ###
[
  {
    "Namespace": "aws:autoscaling:launchconfiguration",
    "OptionName": "IamInstanceProfile",
    "Value": "aws-elasticbeanstalk-ec2-role"
  }
```

```
]
```

Graviton arm64 第一波环境的建议

> **Note**
>
> 本节仅适用于部分客户。如果您在 2021 年 11 月 24 日之前创建具有 Graviton 基于 arm64 的实例类型的新环境，则本节中的信息可能适用于您。

Graviton arm64 第一波环境的建议操作

从 2021 年 10 月和 11 月开始，Elastic Beanstalk 开始在某些区域和一些平台版本增加对 Graviton arm64 处理器的支持浪潮。这第一波是在日期为 2021 年 10 月 13 日、10 月 21 日和 11 月 19 日的 AWS Elastic Beanstalk 发布说明中宣布的。如果您当时创建了基于 arm64 的环境，则说明会告知您使用发布说明中提供的自定义 AMI 配置实例。现在提供对 Graviton arm64 的增强支持，Elastic Beanstalk 默认设置最新平台版本中 arm64 实例类型的 AMI。

如果您使用第一波版本中提供的自定义 AMI 创建环境，我们建议您执行以下操作来维护正常运行和工作的环境。

1. 从环境中删除自定义 AMI。
2. 使用最新的平台版本更新环境。
3. 设置托管平台更新以在计划的维护时段内自动升级到最新平台版本。

> **Note**
>
> Elastic Beanstalk 不会自动替换自定义 AMI。您必须在步骤 1 中删除自定义 AMI，以使步骤 2 中的下一次平台更新对其进行更新。

以下程序将指导您完成这些步骤。这些 AWS CLI 示例适用于使用以下信息创建的环境。

```
aws elasticbeanstalk create-environment \
--region us-east-1 \
--application-name my-app \
--environment-name my-env \
```

```
--solution-stack-name "64bit Amazon Linux 2 v3.4.7 running Docker" \
--option-settings \
Namespace=aws:autoscaling:launchconfiguration,OptionName=IamInstanceProfile,Value=aws-elasticbeanstalk-ec2-role \
Namespace=aws:ec2:instances,OptionName=InstanceTypes,Value=t4g.small \
Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId,Value=ami-0fbdb88ce139244bf
```

更新在第一波 Graviton arm64 支持下创建的 arm64 环境

1. 运行 update-environment 以删除自定义 AMI 设置。

   ```
   aws elasticbeanstalk update-environment \
   --region us-east-1 \
   --environment-name my-env \
   --options-to-remove \
   Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId
   ```

2. 使用最新的平台版本更新环境。请选择以下选项之一。

   - 控制台选项 — 使用 Elastic Beanstalk 控制台更新平台版本。有关更多信息，请参阅更新环境的平台版本。
   - AWS CLI 选项— 运行 AWS update-environ ment 命令，指定最新可用的平台版本。

     ```
     aws elasticbeanstalk update-environment \
     --region us-east-1 \
     --environment-name my-env \
     --solution-stack-name "64bit Amazon Linux 2 v3.4.9 running Docker"
     ```

     > **Note**
     >
     > 该list-available-solution-stacks命令提供您的账户在某个 AWS 地区可用的平台版本列表。
     >
     > ```
     > aws elasticbeanstalk list-available-solution-stacks --region us-east-1 --query SolutionStacks
     > ```

3. 使用 Elastic Beanstalk 控制台为您的环境设置托管平台更新。托管平台更新会在计划的维护时段内将您的环境自动升级到最新平台版本。您的应用程序在更新过程中会继续提供服务。有关更多信息，请参阅托管平台更新。

aws:autoscaling:launchconfiguration 命名空间

您可以使用 aws:autoscaling:launchconfiguration 命名空间中的配置选项来配置您的环境的实例，其中包括未在控制台中提供的附加选项。

以下配置文件示例使用本主题中的基本配置选项。例如，它使用IMDS中讨论的 DisableIMDSv1 选项。它还使用安全性中讨论的 EC2KeyName 和 IamInstanceProfile 选项，以及 BlockDeviceMappings 选项，该选项在控制台中不可用。

```
option_settings:
  aws:autoscaling:launchconfiguration:
    SecurityGroups: my-securitygroup
    MonitoringInterval: "1 minute"
    DisableIMDSv1: false
    EC2KeyName: my-keypair
    IamInstanceProfile: "aws-elasticbeanstalk-ec2-role"
    BlockDeviceMappings: "/dev/sdj=:100,/dev/sdh=snap-51eef269,/dev/sdb=ephemeral0"
```

您可以使用 BlockDeviceMappings 来为实例配置其他块储存设备。有关更多信息，请参阅 Amazon EC2 用户指南中的块储存设备映射。

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关详细信息，请参阅 建议值。

在环境实例上配置实例元数据服务

实例元数据是有关 Amazon Elastic Compute Cloud (Amazon EC2) 实例的数据，可供应用程序用来配置或管理正在运行的实例。实例元数据服务（IMDS）是实例上的组件，在实例上进行编码，用于安全访问实例元数据。这些代码可以是您的环境实例上的 Elastic Beanstalk 平台代码 AWS ，也可以是您的应用程序可能正在使用的软件开发工具包，甚至是您的应用程序自己的代码。有关更多信息，请参阅《Amazon EC2 用户指南》中的实例元数据和用户数据。

代码可以使用以下两种方法之一从正在运行的实例访问实例元数据：实例元数据服务版本 1 (IMDSv1) 或实例元数据服务版本 2 (IMDSv2)。IMDSv2 使用面向会话的请求，并防范了多种类型的漏洞，这些漏洞可用于尝试访问 IMDS。有关这两种方法的信息，请参阅 Amazon EC2 用户指南中的配置实例元数据服务。

Sections

- IMDS 平台支持

- 选择 IMDS 方法
- 使用 Elastic Beanstalk 控制台配置 IMDS
- aws:autoscaling:launchconfiguration 命名空间

IMDS 平台支持

较早的 Elastic Beanstalk 平台版本支持 IMDSv1。较新的 Elastic Beanstalk 平台版本（所有 Amazon Linux 2 平台版本）支持 IMDSv1 和 IMDSv2。您可以将环境配置为支持这两种方法（默认）或禁用 IMDSv1。

> **Note**
>
> 禁用 IMDSv1 需要使用 Amazon EC2 启动模板。在创建或更新环境过程中配置此功能时，Elastic Beanstalk 尝试将环境配置为使用 Amazon EC2 启动模板（如果环境尚未使用启动模板）。在这种情况下，如果您的用户策略缺乏必要的权限，则创建或更新环境可能会失败。因此，我们建议您使用托管用户策略，或者将所需的权限添加到自定义策略中。有关所需权限的详细信息，请参阅the section called “创建自定义用户策略”。

选择 IMDS 方法

在确定希望环境支持的 IMDS 方法时，请考虑以下使用案例：

- AWS SDK — 如果您的应用程序使用 S AWS DK，请确保使用最新版本的 SDK。软件开发工具包会调用 IM AWS DS，而较新版本的 SDK 尽可能使用 imdsv2。如果您禁用了 IMDSv1，或如果您的应用程序使用较早的 SDK 版本，则 IMDS 调用可能会失败。
- 您的应用程序代码 — 如果您的应用程序发出 IMDS 调用，请考虑使用 AWS SDK，这样您就可以进行调用，而不是直接发出 HTTP 请求。通过这种方式，您无需进行代码更改即可在 IMDS 方法之间切换。S AWS DK 尽可能使用 imdsv2。
- Elastic Beanstalk 平台代码 — 我们的代码 AWS 通过 SDK 进行 IMDS 调用，因此在所有支持的平台版本上使用 imdsv2。如果您的代码使用 up-to-date AWS SDK 并通过 SDK 进行所有 IMDS 调用，则可以放心地禁用 imdsv1。

使用 Elastic Beanstalk 控制台配置 IMDS

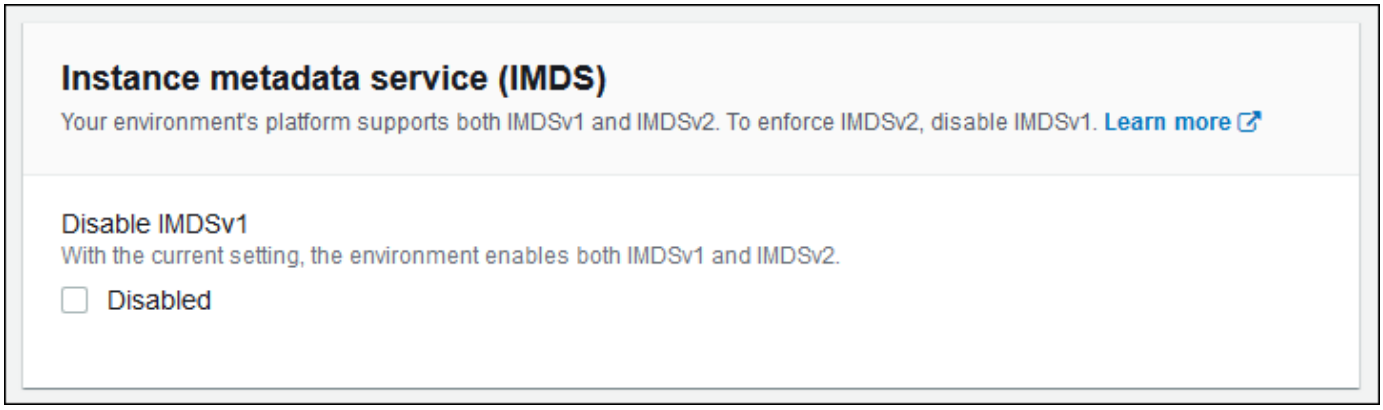
您可以在 Elastic Beanstalk 控制台中修改 Elastic Beanstalk 环境的 Amazon EC2 实例配置。

使用 Elastic Beanstalk 控制台在 Amazon EC2 实例上配置 IMDS

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Instances (实例) 配置类别中，选择 Edit (编辑)。



5. 设置 Disable IMDSv1（禁用 IMDSv1）以强制实施 IMDSv2。清除 Disable IMDSv1（禁用 IMDSv1）可同时启用 IMDSv1 和 IMDSv2。
6. 要保存更改，请选择页面底部的 Apply（应用）。

aws:autoscaling:launchconfiguration 命名空间

您可以使用 `aws:autoscaling:launchconfiguration` 命名空间中的配置选项在环境实例上配置 IMDS。

以下配置文件示例使用 `DisableIMDSv1` 选项禁用 IMDSv1。

```
option_settings:
  aws:autoscaling:launchconfiguration:
    DisableIMDSv1: true
```

Elastic Beanstalk 环境的 Auto Scaling 组

您的 AWS Elastic Beanstalk 环境包括一个 A uto Scaling 组，用于管理您的环境中的 Amazon EC2 实例。在单实例环境中，Auto Scaling 组可确保始终有一个正在运行的实例。在负载均衡的环境中，您将为组配置一系列要运行的实例，Auto Scaling 将根据负载按需添加或删除实例。

Auto Scaling 组还会为您环境中的实例应用启动配置。您可以修改启动配置，以更改实例类型、密钥对、Amazon Elastic Block Store (Amazon EBS) 存储和只能在启动实例时配置的其他设置。

Auto Scaling 组使用两个 Amazon CloudWatch 警报来触发扩展操作。当每个实例的平均出站网络流量在 5 分钟时间段内高于 6 MiB 或低于 2 MiB 时，默认触发器将扩展。要高效使用 Auto Scaling，请根据您的应用程序、实例类型和服务要求配置触发器。您可以基于若干个统计数据 (包括延迟、磁盘 I/O、CPU 使用率和请求计数) 来进行扩展。

要通过可预测的峰值流量期间来优化环境对 Amazon EC2 实例的使用，请配置 Auto Scaling 组以更改计划中的实例计数。您可以安排每天或每周重复一次的组配置更改，或安排一次性更改，以便为可为网站带来大量流量的营销活动做好准备。

作为一个选项，Elastic Beanstalk 可以为您的环境组合按需实例和 Spot 实例。您可以配置 Amazon EC2 Auto Scaling，以通过启用容量再平衡监控和自动响应影响 Spot 实例可用性的更改。

Auto Scaling 监控它启动的每个 Amazon EC2 实例的运行状况。如果任何实例出现意外终止，Auto Scaling 会检测该终止，并启动替代实例。要配置组以使用负载均衡器的运行状况检查机制，请参阅Auto Scaling 运行状况检查设置。

您可以使用 Elastic Beanstalk 控制台、EB CLI 或配置选项为您的环境配置 Auto Scaling。

主题

Spot 实例支持

要利用 Amazon EC2 Spot 实例，您可以为您的环境启用 Spot 选项。然后，您环境的 Auto Scaling 组会将 Amazon EC2 购买选项组合在一起，并将按需实例和 Spot 实例组合在一起。

以下本主题将介绍为您的环境启用 Spot 实例请求的方法：

- Elastic Beanstalk 控制台 — 有关更多信息，请参阅 the section called “使用 Elastic Beanstalk 控制台进行 Auto Scaling 组配置” 中的机群组合 (Fleet composition)。
- EB CLI – 有关更多信息，请参阅 the section called “使用 EB CLI 进行 Auto Scaling 组配置”。
- `aws:ec2:instances` 命名空间配置选项 – 有关更多信息，请参阅 the section called “配置选项”。

> **Important**
>
> 对 Spot 实例的需求在不同时间可能有显著的差异，Spot 实例的可用性也会因为未使用 Amazon EC2 实例的可用数量而差别巨大。Spot 实例可能会中断。
> 为了有助于最大限度地减少这些中断对应用程序的影响，您可以启用 Amazon EC2 Auto Scaling 随附的“容量再平衡”选项。启用此功能后，EC2 会在中断前自动尝试替换 Auto Scaling 组中的 Spot 实例。要启用此功能，请使用 Elastic Beanstalk 控制台以配置 Auto Scaling 组。或者，您可以在 aws:autoscaling:asg 命名空间中将 Elastic Beanstalk `EnableCapacityRebalancing` 配置选项设置为 `true`。
> 有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的容量重新平衡和 Amazon EC2 用户指南中的发现实例中断。

Elastic Beanstalk 提供了多个配置选项来支持 Spot 功能。这些内容将在以下与配置 Auto Scaling 组相关的章节中讨论。

在 aws:ec2:instances 命名空间中，这些选项里有两个值得特别注意：

- `SpotFleetOnDemandBase`
- `SpotFleetOnDemandAboveBasePercentage`

这两个选项与 aws:autoscaling:asg 命名空间中的 `MinSize` 选项相关联。

- 只有 `MinSize` 决定了您环境的初始容量，即您希望运行的最低实例数。

- SpotFleetOnDemandBase 不会影响初始容量。启用 Spot 时，此选项仅确定在考虑任何 Spot 实例前预置的按需实例数。
- 考虑何时 SpotFleetOnDemandBase 小于 MinSize。您仍然会得到完全的 MinSize 实例作为初始容量。至少这些实例中的 SpotFleetOnDemandBase 必须是按需实例。
- 考虑何时 SpotFleetOnDemandBase 大于 MinSize。随着环境的扩展，将保证您至少可获得等于两个值之间的差值的额外实例数。换句话说，可以保证您在满足 SpotFleetOnDemandBase 要求之前至少获得额外的按需 (SpotFleetOnDemandBase - MinSize) 实例。

在生产环境中，将 Spot 实例作为可扩展、负载均衡的环境的一部分尤其有用。我们建议不要在单实例环境中使用 Spot。如果 Spot 实例不可用，您可能会丢失环境的所有容量（单实例）。您仍可以在单实例环境中使用 Spot 实例进行开发或测试。此种情况下，请务必将 SpotFleetOnDemandBase 和 SpotFleetOnDemandAboveBasePercentage 均设置为零。任何其他设置都会导致使用按需实例。

> **注意**
>
> - 一些较旧的 AWS 账户可能会为 Elastic Beanstalk 提供不支持竞价型实例的默认实例类型（例如 t1.micro）。如果启用 Spot 实例请求，并且显示 None of the instance types you specified supports Spot (您指定的所有实例类型均不支持 Spot) 错误，请务必配置支持 Spot 的实例类型。要选择 Spot 实例类型，请使用 Spot Instance Advisor。
> - 启用 Spot 实例请求需要使用 Amazon EC2 启动模板。在创建或更新环境过程中配置此功能时，Elastic Beanstalk 尝试将环境配置为使用 Amazon EC2 启动模板（如果环境尚未使用启动模板）。在这种情况下，如果您的用户策略缺乏必要的权限，则创建或更新环境可能会失败。因此，我们建议您使用托管用户策略，或者将所需的权限添加到自定义策略中。有关所需权限的详细信息，请参阅the section called “创建自定义用户策略”。

以下示例演示了设置各种调整选项的不同方案。所有示例都采用已启用 Spot 实例请求的负载均衡环境。

Example 1：初始容量包含按需实例和 Spot 实例

选项设置

选项	命名空间	值
MinSize	aws:autoscaling:asg	10

选项	命名空间	值
MaxSize	aws:autoscaling:asg	24
SpotFleetOnDemandBase	aws:ec2:instances	4
SpotFleetOnDemandAboveBasePercentage	aws:ec2:instances	50

在此示例中，环境的初始实例数是十个，其中七个是按需实例（四个基本实例，以及基本实例以外的六个实例的 50%），三个是 Spot 实例。该环境最多可扩展到 24 个实例。扩展时，队列中四个基本按需实例以外的按需实例比例保持为 50%，该队列整体最多可包含 24 个实例，其中 14 个是按需实例（4 个基本实例，以及基本实例以外的 20 个实例的 50%），10 个是 Spot 实例。

Example 2：所有按需初始容量

选项设置

选项	命名空间	值
MinSize	aws:autoscaling:asg	4
MaxSize	aws:autoscaling:asg	24
SpotFleetOnDemandBase	aws:ec2:instances	4
SpotFleetOnDemandAboveBasePercentage	aws:ec2:instances	50

在此示例中，环境的初始实例数是四个，所有这些实例都是按需实例。该环境最多可扩展到 24 个实例。扩展时，队列中四个基本按需实例以外的按需实例比例保持为 50%，该队列整体最多可包含 24 个实例，其中 14 个是按需实例（4 个基本实例，以及基本实例以外的 20 个实例的 50%），10 个是 Spot 实例。

Example 3：初始容量以外的额外按需基本实例

选项设置

选项	命名空间	值
MinSize	aws:autoscaling:asg	3
MaxSize	aws:autoscaling:asg	24
SpotFleetOnDemandBase	aws:ec2:instances	4
SpotFleetOnDemandAboveBasePercentage	aws:ec2:instances	50

在此示例中，环境的初始实例数是三个，所有这些实例都是按需实例。该环境最多可扩展到 24 个实例。除开三个初始实例以外添加的第一个实例是按需实例，用于凑齐四个基本按需实例。随着进一步扩展，队列中四个基本按需实例以外的按需实例比例保持为 50%，该队列整体最多可包含 24 个实例，其中 14 个是按需实例（4 个基本实例，以及基本实例以外的 20 个实例的 50%），10 个是 Spot 实例。

使用 Elastic Beanstalk 控制台进行 Auto Scaling 组配置

可以通过在 Elastic Beanstalk 控制台中环境的 Configuration (配置) 页面上编辑 Capacity (容量) 来配置 Auto Scaling 的工作方式。

在 Elastic Beanstalk 控制台中配置 Auto Scaling 组

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在 Auto Scaling group (Auto Scaling 组) 部分中，配置以下设置。

- Environment type (环境类型) - 选择 Load balanced (负载均衡)。
- Min instances (最小实例数) - 组在任何时间均应包含的最小 EC2 实例数。组从最小计数开始，当满足扩展触发条件时，则添加实例。
- Max instances (最大实例数) - 组在任何时间均应包含的最大 EC2 实例数。

> **Note**
>
> 如果您使用滚动更新，请确保最大实例计数高于滚动更新的 Minimum instances in service (使用的最小实例数) 设置。

- 机群组合 (Fleet composition) — 默认值为按需实例。要启用 Spot 实例请求，请选择组合购买选项和实例。

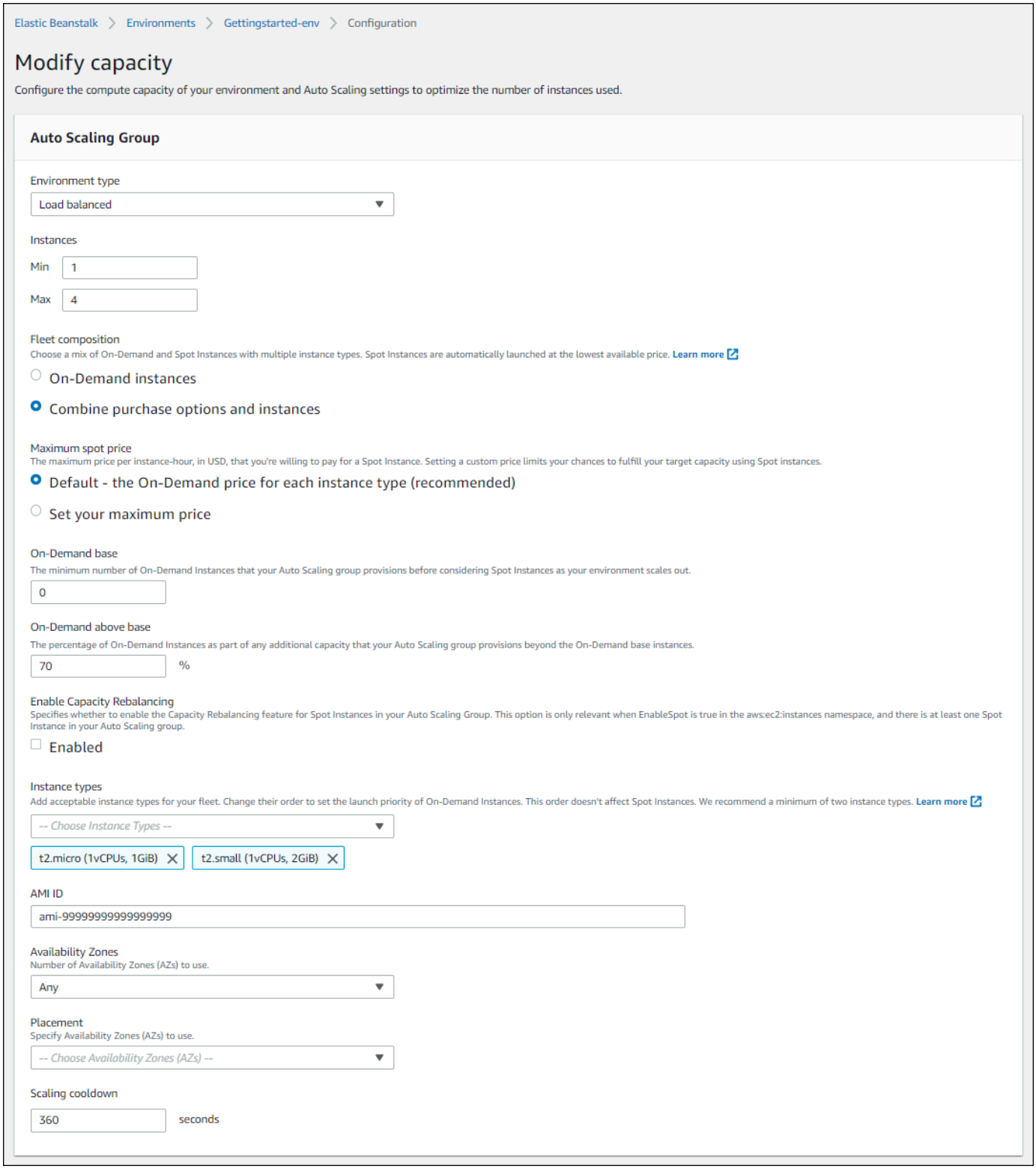
  如果选择启用 Spot 实例请求，请启用以下选项：

  - 最高竞价价格 — 有关竞价型实例最高价格选项的建议，请参阅 Amazon EC2 用户指南中的竞价型实例定价历史记录。
  - 按需基准 (On-Demand base) – 向外扩展环境时，在考虑 Spot 实例之前，Auto Scaling 组预配置的最小按需实例数。
  - 按需上方基准 (On-Demand above base)– Auto Scaling 组在按需基本实例之外调配的任何额外容量中，按需实例所占的百分比。

  > **Note**
  >
  > 选项 On-Demand base（按需基准）和 On-Demand above base（按需上方基准）关联到之前列出的 Min（最小值）和 Max（最大值）实例选项。有关这些选项和例子的更多信息，请参阅 the section called “Spot 实例支持”。

  - 启用容量再平衡— 仅当 Auto Scaling 组中至少有一个 Spot 实例时，此选项才相关。启用此功能后，EC2 会在中断前自动尝试替换 Auto Scaling 组中的 Spot 实例，以最大限度地减少 Spot 实例对应用程序的中断。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的容量再平衡。
- Instance type (实例类型) - 为运行应用程序而启动的 Amazon EC2 实例的类型。有关更多信息，请参阅 the section called “实例类型”。
- AMI ID - Elastic Beanstalk 用来启动环境中的 Amazon EC2 实例的计算机映像。有关更多信息，请参阅 the section called “AMI ID”。

- Availability Zones (可用区) - 选择环境实例要跨越的可用区的数量。默认情况下，Auto Scaling 组会在所有可用区中均匀启动实例。要将实例集中在少数几个区域中，请选择要使用的区域数。对于生产环境而言，至少要使用两个区域，以确保当一个可用区中断服务时您的应用程序仍然可用。
- Placement (放置) (可选) - 选择要使用的可用区。如果您的实例需要连接至特定区域中的资源或者您购买了区域特定的预留实例，请使用此设置。如果您在自定义 VPC 中启动环境，则无法配置此选项。在自定义 VPC 中，您需要为分配给环境的子网选择可用区。
- Scaling cooldown (扩展冷却时间) - 在扩展之后、在继续评估触发器之前等待实例启动或终止的时间 (以秒为单位)。有关更多信息，请参阅扩展冷却时间。



6. 要保存更改，请选择页面底部的 Apply（应用）。

使用 EB CLI 进行 Auto Scaling 组配置

使用 eb create 命令创建环境时，可以指定几个与环境的 Auto Scaling 组相关的选项。这些选项可帮助您控制环境的容量。

`--single`

使用一个 Amazon EC2 实例创建环境，并且不使用负载均衡器。如果您未使用此选项，负载均衡器会被添加到创建的环境中。

`--enable-spot`

为您的环境启用 Spot 实例请求。

只有 eb create 命令的以下这些选项才能与 `--enable-spot` 一起使用。

`--instance-types`

列出您希望环境使用的 Amazon EC2 实例类型。

`--spot-max-price`

您愿意为 Spot 实例支付的每单位小时的最高价（美元）。有关竞价型实例最高价格选项的建议，请参阅 Amazon EC2 用户指南中的竞价型实例定价历史记录。

`--on-demand-base-capacity`

扩展环境时，在考虑 Spot 实例之前，Auto Scaling 组预配置的最小按需实例数。

`--on-demand-above-base-capacity`

Auto Scaling 组在 `--on-demand-base-capacity` 选项指定的超过的实例数作为额外容量预配置的按需实例的百分比。

以下示例创建一个环境，并配置 Auto Scaling 组以便为新环境启用 Spot 实例请求。在此示例中，可以使用这三种实例类型。

```
$ eb create --enable-spot --instance-types "t2.micro,t3.micro,t3.small"
```

> **Important**
>
> 还有另一个名称类似的选项，称为 `--instance-type`（无“s”），EB CLI 仅在处理按需实例时识别该选项。请勿将 `--instance-type`（无“s”）与 `--enable-spot` 选项一同使用。如

果您这样做，EB CLI 将忽略它。请改为将 `--instance-types`（带有“s”）与 `--enable-spot` 选项一同使用。

配置选项

Elastic Beanstalk 在以下两个命名空间中提供了用于 Auto Scaling 设置的配置选项：`aws:autoscaling:asg` 和 `aws:ec2:instances`。

aws:autoscaling:asg 命名空间

`aws:autoscaling:asg` 命名空间提供了用于整体调整和可用性的选项。

以下配置文件示例配置 Auto Scaling 组以便使用两到四个实例、特定可用区和 12 分钟（720 秒）的冷却时间。已启用 Spot 实例的容量再平衡。如下面的配置文件示例所示，如果在 `aws:ec2:instances` 命名空间中将 `EnableSpot` 设置为 `true`，则只有最后一个选项才有效。

```
option_settings:
  aws:autoscaling:asg:
    Availability Zones: Any
    Cooldown: '720'
    Custom Availability Zones: 'us-west-2a,us-west-2b'
    MaxSize: '4'
    MinSize: '2'
    EnableCapacityRebalancing: true
```

aws:ec2:instances 命名空间

`aws:ec2:instances` 命名空间提供与环境的实例相关的选项，包括 Spot 实例管理。它是 `aws:autoscaling:launchconfiguration` 和 `aws:autoscaling:asg` 的补充。

当您更新环境配置并从 `InstanceTypes` 选项中删除一个或多个实例类型时，Elastic Beanstalk 会终止在任何已删除的实例类型上运行的任何 Amazon EC2 实例。然后，您环境的 Auto Scaling 组根据需要启动新实例，以使用当前指定的实例类型来完成所需的容量。

以下配置文件示例配置 Auto Scaling 组，以便为环境启用 Spot 实例请求。在此示例中，使用三种可能的实例类型。至少采用一个按需实例作为基准容量，并采用 33% 的按需实例作为补充用的任何额外容量。

```
option_settings:
```

```
  aws:ec2:instances:
    EnableSpot: true
    InstanceTypes: 't2.micro,t3.micro,t3.small'
    SpotFleetOnDemandBase: '1'
    SpotFleetOnDemandAboveBasePercentage: '33'
```

要选择 Spot 实例类型，请使用 Spot Instance Advisor。

Auto Scaling 触发

Elastic Beanstalk 环境中的 Auto Scaling 组使用两个 Amazon CloudWatch 警报来触发扩展操作。当每个实例的平均出站网络流量在 5 分钟时间段内高于 6 MB 或低于 2 MB 时，默认触发器将扩展。要高效使用 Amazon EC2 Auto Scaling，请根据您的应用程序、实例类型和服务要求配置触发器。您可以基于若干个统计数据 (包括延迟、磁盘 I/O、CPU 使用率和请求计数) 来进行扩展。

有关 CloudWatch 指标和警报的更多信息，请参阅 Amazon CloudWatch 用户指南 中的 Amazon CloudWatch 概念。

配置 Auto Scaling 触发器

您可以配置触发器，以便在 Elastic Beanstalk 控制台中调整环境的 Auto Scaling 组中的实例数。

在 Elastic Beanstalk 控制台中配置触发器

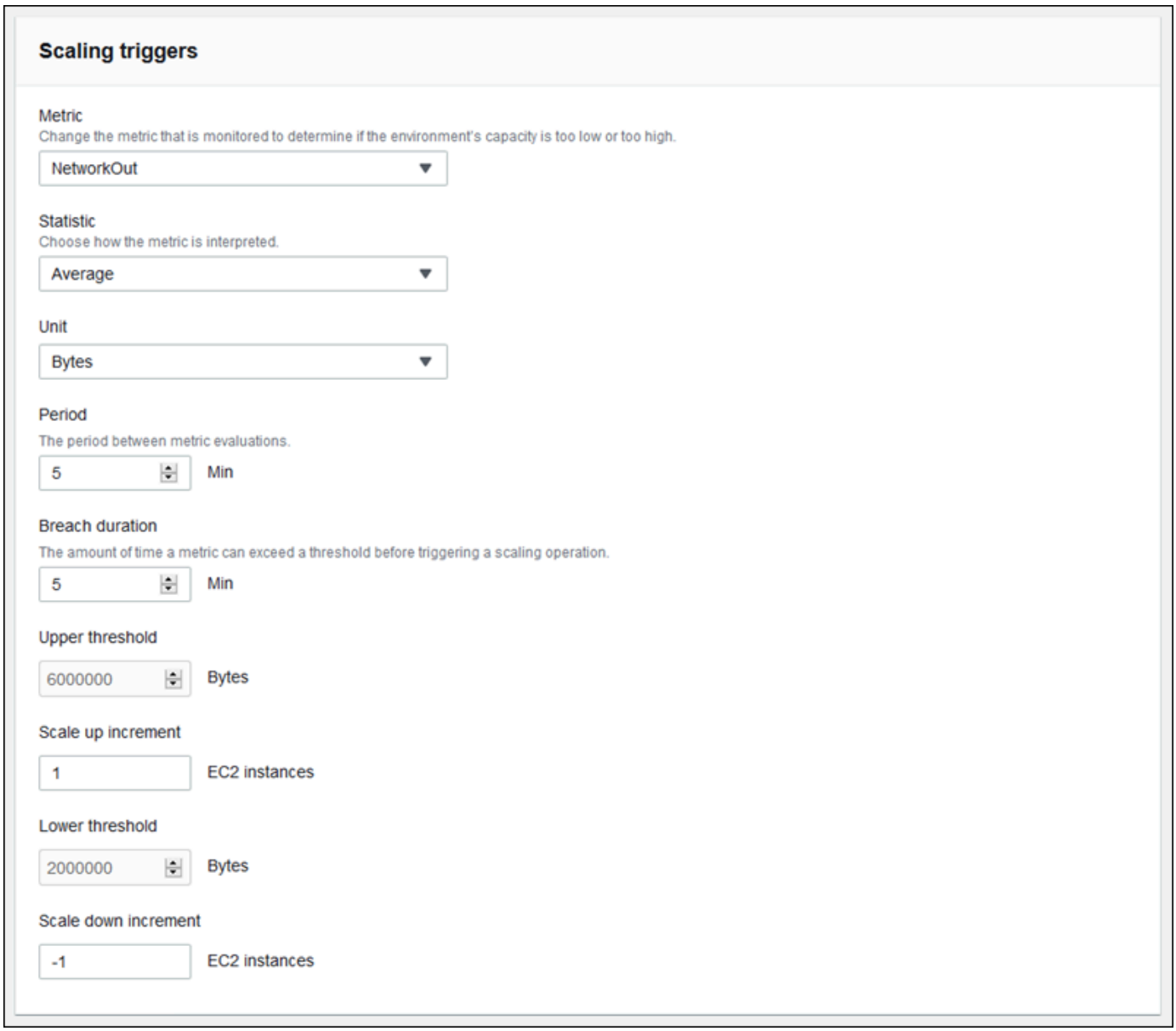
1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在扩展触发部分中，配置以下设置：

   - Metric (指标) – 用于 Auto Scaling 触发器的指标。
   - Statistic (统计数据) - 触发器应使用的统计数据计算，如 `Average`。
   - Unit (单位) - 触发器指标的单位，例如 Bytes (字节)。
   - Period (周期) – 指定 Amazon CloudWatch 测量触发指标的频率。

- Breach duration (违例持续时间) - 触发扩展操作前，指标可以超出上限阈值和下限阈值的时间（以分钟为单位）。
- Upper threshold (上限) - 如果指标超出该违例持续时间数值，则会触发扩展操作。
- Scale up increment (扩展增量) - 执行扩展活动时要添加的 Amazon EC2 实例数。
- Lower threshold (下限) - 如果指标小于该违例持续时间值，则会触发扩展操作。
- Scale down increment (缩减增量) - 执行扩展活动时要删除的 Amazon EC2 实例数。



6. 要保存更改，请选择页面底部的 Apply（应用）。

aws:autoscaling:trigger 命名空间

Elastic Beanstalk 为 `aws:autoscaling:trigger` 命名空间中的 Auto Scaling 设置提供配置选项。此命名空间中的设置按它们所适用的资源来组织。

```
option_settings:
  AWSEBAutoScalingScaleDownPolicy.aws:autoscaling:trigger:
    LowerBreachScaleIncrement: '-1'
  AWSEBAutoScalingScaleUpPolicy.aws:autoscaling:trigger:
    UpperBreachScaleIncrement: '1'
  AWSEBCloudwatchAlarmHigh.aws:autoscaling:trigger:
    UpperThreshold: '6000000'
  AWSEBCloudwatchAlarmLow.aws:autoscaling:trigger:
    BreachDuration: '5'
    EvaluationPeriods: '1'
    LowerThreshold: '2000000'
    MeasureName: NetworkOut
    Period: '5'
    Statistic: Average
    Unit: Bytes
```

计划的 Auto Scaling 操作

要通过可预测的峰值流量期间来优化环境对 Amazon EC2 实例的使用，请配置 Amazon EC2 Auto Scaling 组以更改计划中的实例计数。您可以用重复操作配置您的环境，以便在上午扩展，在夜间流量低时缩减。例如，如果您的营销活动会在有限时间段内使网站的流量增加，则可计划一个一次性事件以便在活动开始时扩展，而计划另一个事件以便在活动结束时缩减。

每个环境最多可以定义 120 个活跃的计划操作。Elastic Beanstalk 还会保留最多 150 个过期的计划操作，可以通过更新设置来重用这些操作。

配置计划的操作

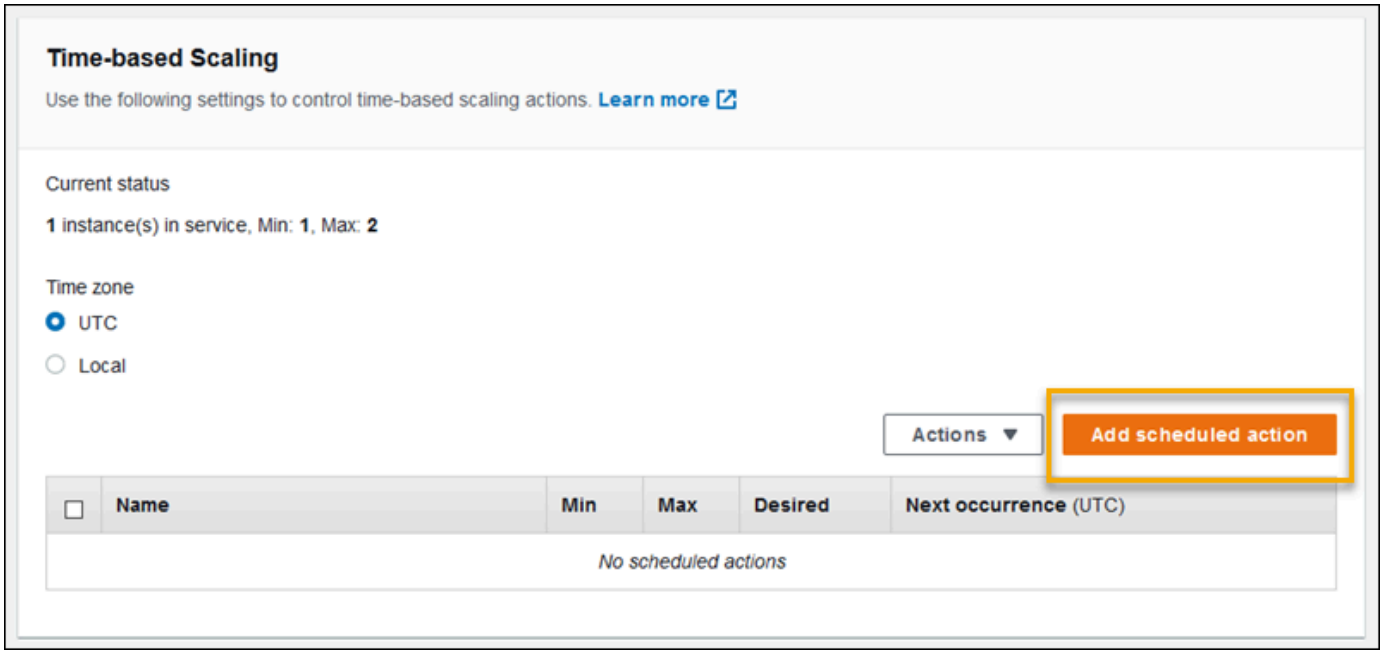
您可以在 Elastic Beanstalk 控制台中为环境的 Auto Scaling 组创建计划的操作。

在 Elastic Beanstalk 控制台中配置计划的操作

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 在 Time-based scaling (基于时间的扩展) 部分中，选择 Add scheduled action (添加计划的操作)。



6. 填写以下计划操作设置：

   - Name (名称) - 指定一个唯一的名称，最多包含 255 个字母数字字符且不带空格。
   - Instances (实例) - 选择要应用于 Auto Scaling 组的最小和最大实例计数。
   - Desired capacity (所需容量) (可选) - 为 Auto Scaling 组设置所需的初始容量。在应用计划的操作后，触发器将根据其设置调整所需容量。
   - Occurrence (出现) - 选择 Recurring (定期) 以在计划中重复扩展操作。
   - Start time (开始时间) - 对于一次性操作，请选择运行操作的日期和时间。

     对于重复操作，开始时间是可选的。指定它以选择执行操作的最早时间。在此时间之后，操作将根据循环表达式重复发生。
   - Recurrence (循环) - 使用 Cron 表达式指定您希望计划操作发生的频率。例如，`30 6 * * 2` 在 UTC 时间每周二的早上 6:30 运行操作。

- End time (结束时间) (可选) - 对于重复性操作是可选的。如果指定此项，操作将根据循环表达式重新进行，并且在此时间之后不再执行。

  当计划的操作结束时，Auto Scaling 不会自动恢复为其以前的设置。配置第二个计划操作，以根据需要将 Auto Scaling 返回原始设置。

7. 选择 Add。
8. 要保存更改，请选择页面底部的 Apply (应用)。

> **Note**
>
> 计划的操作在应用之前不保存。

aws:autoscaling:scheduledaction 命名空间

如果您需要配置大量计划操作，则可以使用配置文件或 Elastic Beanstalk API 应用来自 YAML 或 JSON 文件的配置选项更改。通过这些方法还可以访问 `Suspend` 选项，以临时停用某个重复的计划操作。

> **Note**
>
> 在控制台之外使用计划操作配置选项时，请使用 ISO 8601 时间格式指定开始时间和结束时间 (采用 UTC)。例如，2015-04-28T04:07:02Z。有关 ISO 8601 时间格式的更多信息，请参阅日期和时间格式。日期在所有计划操作中都必须是唯一的。

Elastic Beanstalk 在 `aws:autoscaling:scheduledaction` 命名空间中提供了用于计划操作设置的配置选项。使用 `resource_name` 字段可指定计划操作的名称。

Example Scheduled-scale-up-specific-time-long.config

此配置文件指示 Elastic Beanstalk 在 2015-12-12T00:00:00Z 从 5 个实例扩展到 10 个实例。

```
option_settings:
  - namespace: aws:autoscaling:scheduledaction
    resource_name: ScheduledScaleUpSpecificTime
    option_name: MinSize
    value: '5'
  - namespace: aws:autoscaling:scheduledaction
```

```
    resource_name: ScheduledScaleUpSpecificTime
    option_name: MaxSize
    value: '10'
  - namespace: aws:autoscaling:scheduledaction
    resource_name: ScheduledScaleUpSpecificTime
    option_name: DesiredCapacity
    value: '5'
  - namespace: aws:autoscaling:scheduledaction
    resource_name: ScheduledScaleUpSpecificTime
    option_name: StartTime
    value: '2015-12-12T00:00:00Z'
```

Example Scheduled-scale-up-specific-time.config

要在 EB CLI 或配置文件中使用简写语法，请将资源名称添加到命名空间。

```
option_settings:
  ScheduledScaleUpSpecificTime.aws:autoscaling:scheduledaction:
    MinSize: '5'
    MaxSize: '10'
    DesiredCapacity: '5'
    StartTime: '2015-12-12T00:00:00Z'
```

Example Scheduled-scale-down-specific-time.config

此配置文件指示 Elastic Beanstalk 在 2015-12-12T07:00:00Z 进行缩减。

```
option_settings:
  ScheduledScaleDownSpecificTime.aws:autoscaling:scheduledaction:
    MinSize: '1'
    MaxSize: '1'
    DesiredCapacity: '1'
    StartTime: '2015-12-12T07:00:00Z'
```

Example Scheduled-periodic-scale-up.config

此配置文件指示 Elastic Beanstalk 在每天上午 9 点进行横向扩展。计划该操作从 2015 年 5 月 14 日开始并于 2016 年 1 月 12 日结束。

```
option_settings:
  ScheduledPeriodicScaleUp.aws:autoscaling:scheduledaction:
```

```
    MinSize: '5'
    MaxSize: '10'
    DesiredCapacity: '5'
    StartTime: '2015-05-14T07:00:00Z'
    EndTime: '2016-01-12T07:00:00Z'
    Recurrence: 0 9 * * *
```

Example Scheduled-periodic-scale-down.config

此配置文件指示 Elastic Beanstalk 在每天下午 6 点进行缩减以便不运行实例。如果您知道应用程序在工作时间之外的大多数时间闲置，则可以创建一个类似的计划操作。如果您的应用程序必须在工作时间之外停止运行，请将 MaxSize 更改为 0。

```
option_settings:
  ScheduledPeriodicScaleDown.aws:autoscaling:scheduledaction:
    MinSize: '0'
    MaxSize: '1'
    DesiredCapacity: '0'
    StartTime: '2015-05-14T07:00:00Z'
    EndTime: '2016-01-12T07:00:00Z'
    Recurrence: 0 18 * * *
```

Example Scheduled-weekend-scale-down.config

此配置文件指示 Elastic Beanstalk 在每个星期五的下午 6 点进行缩减。如果您获知应用程序在周末并未接收到足够多的流量，则可创建一个类似的计划操作。

```
option_settings:
  ScheduledWeekendScaleDown.aws:autoscaling:scheduledaction:
    MinSize: '1'
    MaxSize: '4'
    DesiredCapacity: '1'
    StartTime: '2015-12-12T07:00:00Z'
    EndTime: '2016-01-12T07:00:00Z'
    Recurrence: 0 18 * * 5
```

Auto Scaling 运行状况检查设置

Amazon EC2 Auto Scaling 监控其启动的每个 Amazon Elastic Compute Cloud (Amazon EC2) 实例的运行状况。如果任何实例出现意外终止，Auto Scaling 会检测该终止，并启动替代实例。默认情况

下，为您的环境创建的 Auto Scaling 组使用 Amazon EC2 状态检查。如果环境中的某个实例未通过 Amazon EC2 状态检查，Auto Scaling 将中断该实例并替换它。

Amazon EC2 状态检查只包括实例的运行状况，而不包括应用程序的运行状况或者实例上运行的任意 Docker。如果应用程序崩溃，但它运行所在的实例仍然运行状况良好，则可能会从负载均衡器中逐出该实例，但 Auto Scaling 不会自动替换它。默认行为适合用于故障排除。如果 Auto Scaling 在应用程序崩溃之后立即替换该实例，您可能不会意识到有任何问题，即使它在启动之后快速崩溃。

如果您希望 Auto Scaling 替换其应用程序停止响应的实例，可以使用配置文件配置 Auto Scaling 组来使用 Elastic Load Balancing 运行状况检查。以下示例将组设置为使用负载均衡器的运行状况检查以及 Amazon EC2 状态检查，以确定实例的运行状况。

Example .ebextensions/autoscaling.config

```
Resources:
  AWSEBAutoScalingGroup:
    Type: "AWS::AutoScaling::AutoScalingGroup"
    Properties:
      HealthCheckType: ELB
      HealthCheckGracePeriod: 300
```

有关 `HealthCheckType` 和 `HealthCheckGracePeriod` 属性的更多信息，请参阅 AWS CloudFormation 用户指南中的 AWS::AutoScaling::AutoScalingGroup 和 Amazon EC2 Auto Scaling 用户指南中的 Auto Scaling 实例的运行状况检查。

默认情况下，Elastic Load Balancing 运行状况检查配置为尝试通过端口 80 与您的实例进行 TCP 连接。这可以确认运行在实例上的 Web 服务器接受连接。但您可能希望自定义负载均衡器运行状况检查以确保应用程序处于良好状态，而不仅仅是 Web 服务器处于良好状态。宽限期设置可设置实例在不终止和替换的情况下无法通过运行状况检查的秒数。实例在从负载均衡器中被逐出后仍然能够恢复，因此请为实例指定适合于您的应用程序的时间量。

Elastic Beanstalk 环境的负载均衡器

负载均衡器在环境的实例之间分配流量。启用负载均衡后，AWS Elastic Beanstalk 会创建专用于您的环境的 Elastic Load Balancing 负载均衡器。Elastic Beanstalk 全面管理此负载均衡器，负责安全设置，并在终止环境时终止负载均衡器。

或者，您可以选择跨多个 Elastic Beanstalk 环境共享负载均衡器。使用共享的负载均衡器，您可以避免为每个环境设置专用负载均衡器，从而节省运营成本。您还要为您的环境使用的共享负载均衡器承担更多的管理责任。

Elastic Load Balancing 支持以下负载均衡器类型：

- 经典负载均衡器 – 上一代负载均衡器。将 HTTP、HTTPS 或 TCP 请求流量路由到环境实例上的不同端口。
- Application Load Balancer – 应用层负载均衡器。根据请求路径，将 HTTP 或 HTTPS 请求流量路由到环境实例上的不同端口。
- Network Load Balancer – 网络层负载均衡器。将 TCP 请求流量路由到环境实例上的不同端口。支持主动和被动运行状况检查。

Elastic Beanstalk 支持所有三种负载均衡器类型。下表显示可以与两种使用模式一起使用的类型：

负载均衡器类型	专用	共享
经典负载均衡器	✓ 是	X 否
Application Load Balancer	✓ 是	✓ 是
Network Load Balancer	✓ 是	X 否

> **Note**
>
> 在创建环境控制台向导中禁用了经典负载均衡器（CLB）选项。如果某个现有的环境已经配置了经典负载均衡器，则可以使用 Elastic Beanstalk 控制台或 EB CLI 克隆现有环境，从而创建新环境。您还可以使用 EB CLI 或 AWS CLI 创建配置了经典负载均衡器的新环境。这些命令行工具将使用 CLB 创建一个新环境，即使您的账户中尚不存在该环境。

默认情况下，当您使用 Elastic Beanstalk 控制台或 EB CLI 启用负载均衡时，Elastic Beanstalk 会为您的环境创建 Application Load Balancer。它将负载均衡器配置为侦听端口 80 上的 HTTP 流量，并将该流量转发到同一端口上的实例。您只能在创建环境期间选择您的环境使用的负载均衡器类型。稍后，您可以更改设置以管理运行环境的负载均衡器行为，但不能更改其类型。

> **Note**
>
> 您的环境所在的 VPC 必须至少有位于两个可用区中的子网，才能创建 Application Load Balancer。所有新AWS账户都包含满足此要求的默认 VPC。

请参阅以下主题以了解 Elastic Beanstalk 支持的每个负载均衡器类型、其功能和如何在 Elastic Beanstalk 环境中对其进行配置和管理，以及如何配置负载均衡器以将访问日志上传到 Amazon S3。

主题

配置经典负载均衡器

在启用负载均衡时，将为您的 AWS Elastic Beanstalk 环境配备 Elastic Load Balancing 负载均衡器，以便在您的环境中的实例之间分配流量。Elastic Load Balancing 支持多种负载均衡器类型。要了解更多信息，请参阅 Elastic Load Balancing 用户指南。Elastic Beanstalk 可以为您创建负载均衡器，或者让您指定已创建的共享负载均衡器。

本主题介绍 Elastic Beanstalk 创建并专用于您的环境的 经典负载均衡器 的配置。有关配置 Elastic Beanstalk 支持的所有负载均衡器类型的信息，请参阅Elastic Beanstalk 环境的负载均衡器。

> **Note**
>
> 您只能在创建环境期间选择您的环境使用的负载均衡器类型。稍后，您可以更改设置以管理运行环境的负载均衡器行为，但不能更改其类型。

简介

经典负载均衡器 是 Elastic Load Balancing 的上一代负载均衡器。它支持将 HTTP、HTTPS 或 TCP 请求流量路由到环境实例上的不同端口。

当您的环境使用经典负载均衡器时，默认情况下，Elastic Beanstalk 会将其配置为侦听端口 80 上的 HTTP 流量并转发给同一端口上的实例。尽管您无法删除端口 80 的默认侦听器，但可以将其禁用，它通过阻止流量来实现相同的功能。请注意，您可以添加或删除其他监听器。要支持安全的连接，您可以使用端口 443 上的侦听器和 TLS 证书来配置负载均衡器。

负载均衡器使用运行状况检查确定运行您的应用程序的 Amazon EC2 实例是否正常运行。运行状况检查会按设置的时间间隔对指定的 URL 发出请求。如果 URL 返回错误消息，或者在指定的超时期间内无法返回任何消息，则表示运行状况检查失败。

如果您的应用程序通过满足来自单一服务器上的同一客户端的多个请求来提高性能，您可以将负载均衡器配置为使用粘性会话。利用粘性会话，负载均衡器可将 Cookie 添加到用于标识满足请求的 Amazon EC2 实例的 HTTP 响应。当从同一客户端收到后续请求时，负载均衡器会使用 Cookie 将该请求发送到同一实例。

借助跨区域负载均衡，经典负载均衡器的每个负载均衡器节点会跨所有启用的可用区中的已注册实例平均分配请求。如果禁用了跨区域负载均衡，则每个负载均衡器节点会仅在其可用区中的已注册实例之间平均分配请求。

如果由于运行状况不佳或缩减环境规模而从负载均衡器中删除一个实例，Connection Draining 将为该实例留出一些时间以完成请求，然后再关闭实例和负载均衡器之间的连接。您可以更改为实例预留的用于发送响应的时间量，或完全禁用连接耗尽。

> **Note**
>
> 在您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境时，默认情况下将启用 Connection Draining。对于其他客户端，您可以使用配置选项启用该功能。

您可以使用高级负载均衡器设置在任意端口上配置侦听器，修改其他粘性会话设置，以及配置负载均衡器以安全地连接到 EC2 实例。这些设置是通过配置选项提供的，您可以使用源代码中的配置文件设置这些选项，也可以使用 Elastic Beanstalk API 直接在环境中设置这些选项。在 Elastic Beanstalk 控制台中也提供了其中的很多设置。此外，您还可以将负载均衡器配置为将访问日志上传到 Amazon S3。

使用 Elastic Beanstalk 控制台配置经典负载均衡器

在创建环境期间或以后运行您的环境时，您可以使用 Elastic Beanstalk 控制台配置经典负载均衡器的端口、HTTPS 证书和其他设置。

> **Note**
>
> 在创建环境控制台向导中禁用了经典负载均衡器（CLB）选项。如果某个现有的环境已经配置了经典负载均衡器，则可以使用 Elastic Beanstalk 控制台或 EB CLI 克隆现有环境，从而创建新环境。您还可以使用 EB CLI 或 AWS CLI 创建配置了经典负载均衡器的新环境。这些命令行工具将使用 CLB 创建一个新环境，即使您的账户中尚不存在该环境。

在 Elastic Beanstalk 控制台中配置正在运行环境的经典负载均衡器

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。

> **Note**
>
> 如果 Load balancer (负载均衡器) 配置类别没有 Edit (编辑) 按钮，则表示您的环境没有负载均衡器。要了解如何设置负载均衡器，请参阅更改环境类型。

5. 根据环境需要进行经典负载均衡器配置更改。
6. 要保存更改，请选择页面底部的 Apply（应用）。

经典负载均衡器设置

- 侦听器
- 会话
- 跨可用区负载均衡
- 连接耗尽
- 运行状况检查

侦听器

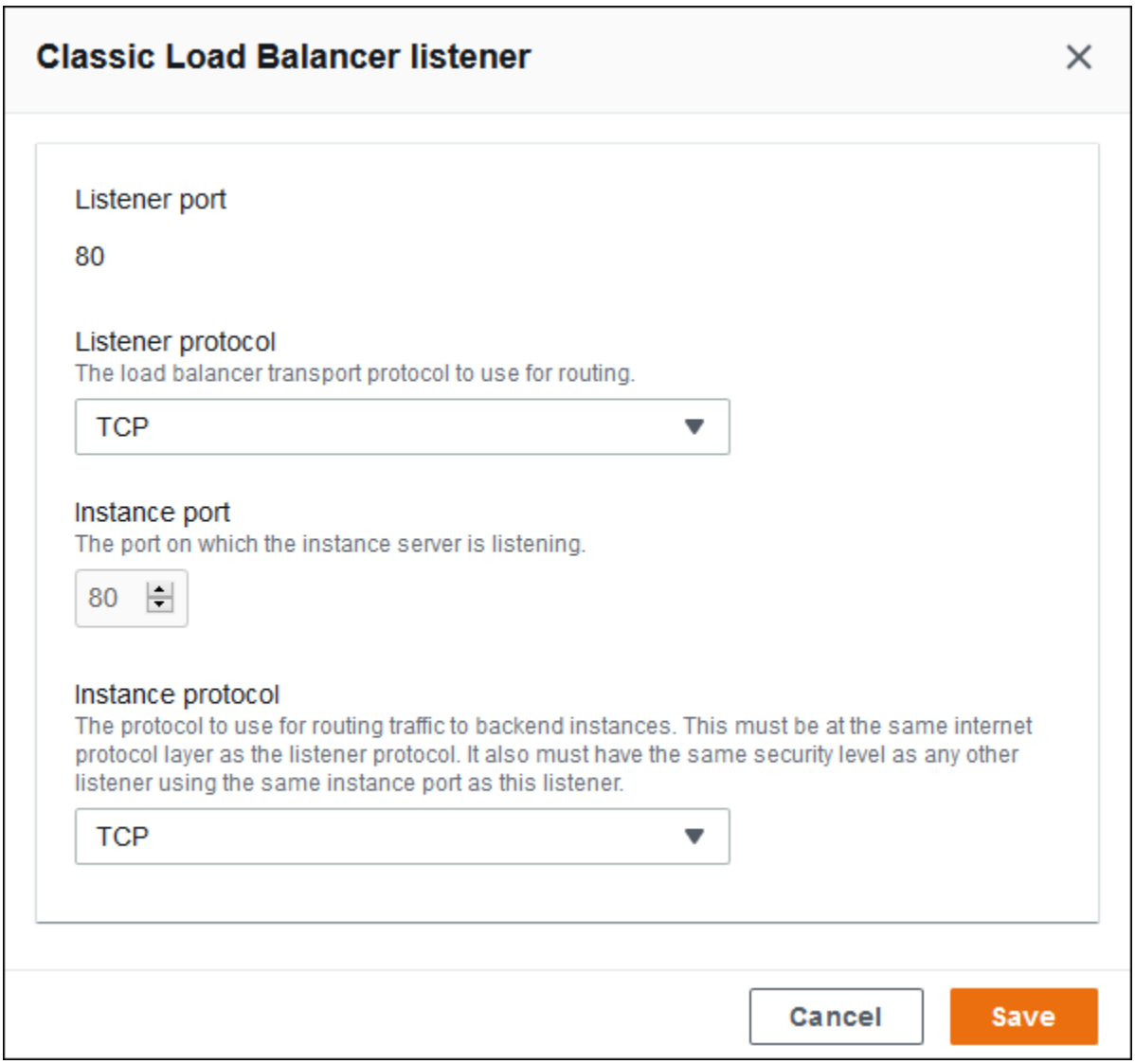
可以使用该列表为您的负载均衡器指定侦听器。每个侦听器使用指定协议将指定端口上的传入客户端流量路由到您的实例。最初，该列表显示默认侦听器，它将端口 80 上的传入 HTTP 流量路由到侦听端口 80 上的 HTTP 流量的环境实例服务器。

> **Note**
>
> 尽管您无法删除端口 80 的默认侦听器，但可以将其禁用，它通过阻止流量来实现相同的功能。

配置现有的侦听器

1. 选中表条目旁边的复选框，选择操作，然后选择所需的操作。
2. 如果您选择编辑，请使用 经典负载均衡器侦听器 对话框编辑设置，然后选择保存。

例如，您可以编辑默认侦听器并将协议从 HTTP 更改为 TCP（如果您希望负载均衡器按原样转发请求）。这可防止负载均衡器重写标头 (包括 `X-Forwarded-For`)。该方法不适用于粘性会话。



添加侦听器

1. 选择添加侦听器。
2. 在 经典负载均衡器侦听器 对话框中，配置所需的设置，然后选择添加。

添加安全侦听器是一个常见案例。下图中的示例为端口 443 上的 HTTPS 流量添加侦听器。该侦听器将传入流量路由到侦听端口 443 上的 HTTPS 流量的环境实例服务器。

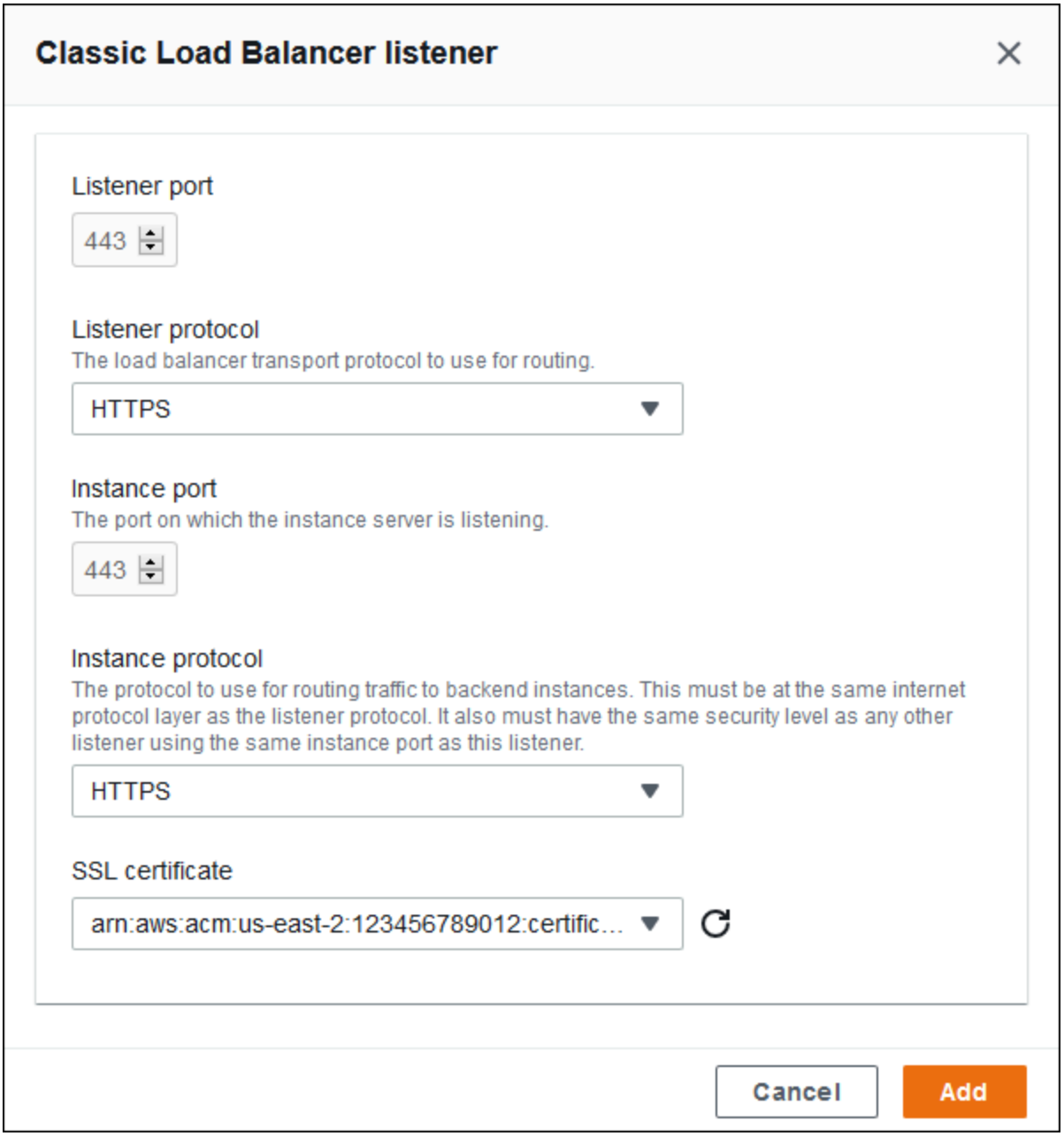
确保您具有有效的 SSL 证书，然后才能配置 HTTPS 侦听器。请执行以下操作之一：

- 如果 AWS Certificate Manager (ACM) 在您的AWS区域中可用，请使用 ACM 创建或导入证书。有关请求 ACM 证书的更多信息，请参阅 AWS Certificate Manager 用户指南中的请求证书。有关将第三方证书导入 ACM 中的更多信息，请参阅 AWS Certificate Manager 用户指南中的导入证书。

- 如果 ACM 在您的 AWS 区域中不可用，请将现有证书和密钥上传到 IAM。有关创建证书并将证书上传到 IAM 的更多信息，请参阅《IAM 用户指南》中的使用服务器证书。

有关在 Elastic Beanstalk 中配置 HTTPS 和使用证书的更多详细信息，请参阅为 Elastic Beanstalk 环境配置 HTTPS。

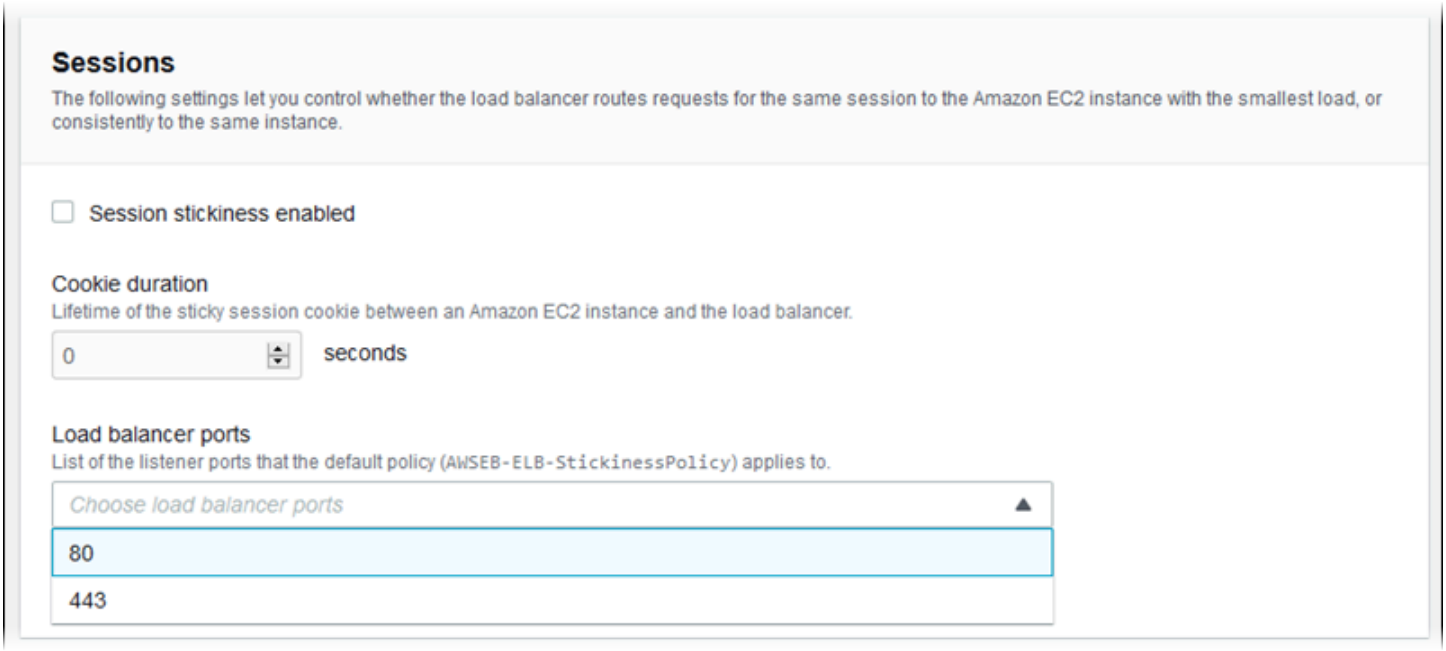
对于 SSL 证书，选择 SSL 证书的 ARN。例如，`arn:aws:iam::123456789012:server-certificate/abc/certs/build` 或 `arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678`。



有关在 Elastic Beanstalk 中配置 HTTPS 和使用证书的详细信息，请参阅为 Elastic Beanstalk 环境配置 HTTPS。

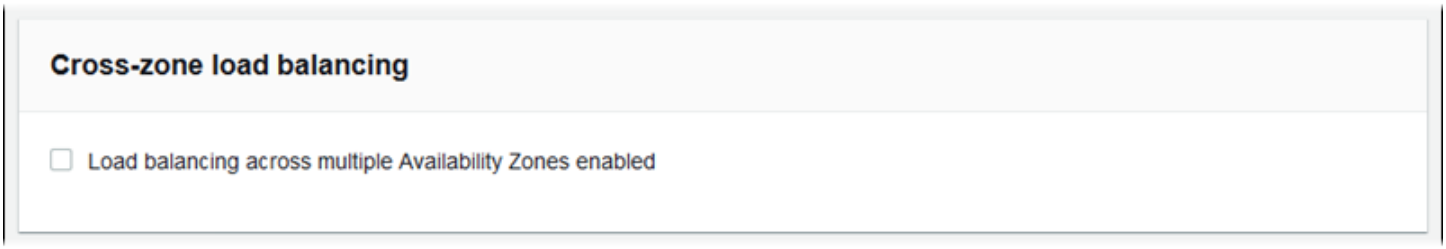
会话

选中或清除会话粘性已启用框来启用或禁用粘性会话。使用 Cookie 持续时间配置粘性会话的持续时间，最多为 **1000000** 秒。在 Load balancer ports (负载均衡器端口) 列表上，选择默认策略 (AWSEB-ELB-StickinessPolicy) 应用于的侦听器端口。



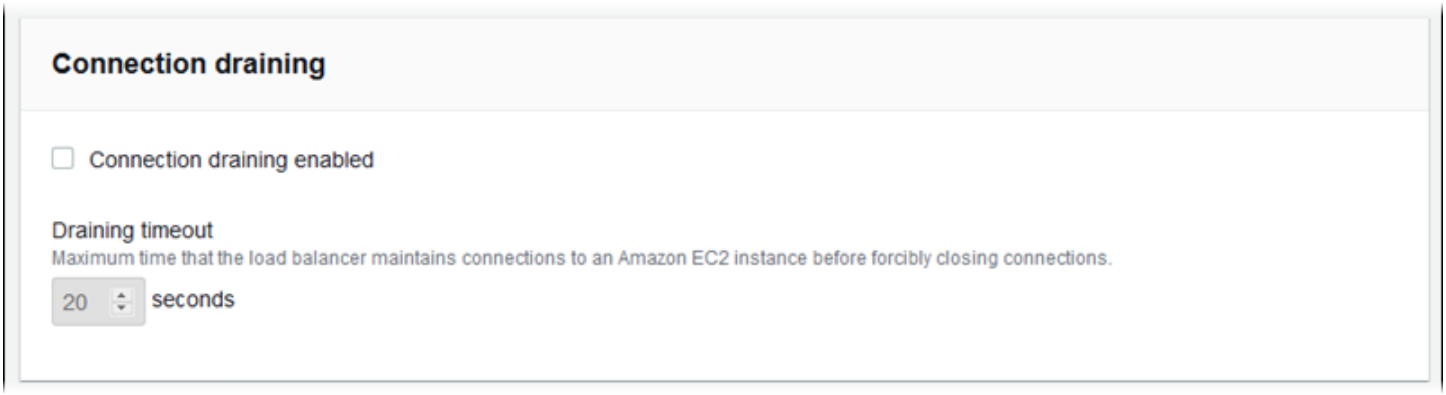
跨可用区负载均衡

选中或清除跨多可用区的负载均衡已启用框以启用或禁用跨区域负载均衡。



连接耗尽

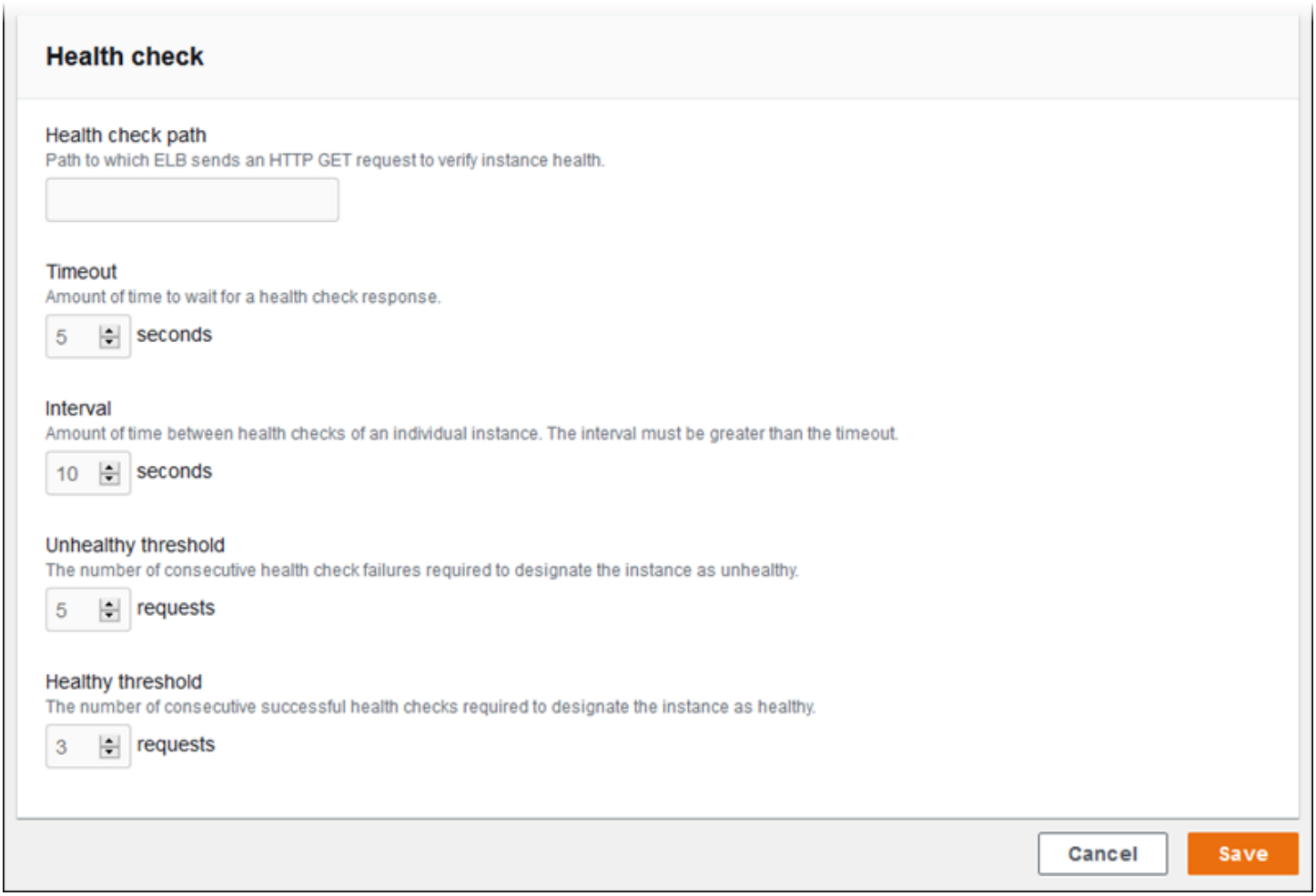
选中或清除连接耗尽已启用框来启用或禁用连接耗尽。设置耗尽超时，最多为 **3600** 秒。



运行状况检查

可以使用以下设置配置负载均衡器运行状况检查：

- Health check path (运行状况检查路径) – 负载均衡器将运行状况检查请求发送到的路径。如果未设置路径，负载均衡器将尝试在端口 80 上建立 TCP 连接以验证运行状况。
- Timeout (超时) – 等待运行状况检查响应的时间（秒）。
- Interval (间隔) – 单个实例的两次运行状况检查间隔的时间（秒）。间隔必须大于超时。
- Unhealthy threshold (不正常阈值) 和 Healthy threshold (正常阈值) – 在 Elastic Load Balancing 更改实例的运行状况之前，实例必须通过或未通过的运行状况检查次数。



> **Note**
>
> Elastic Load Balancing 运行状况检查不会影响环境的 Auto Scaling 组的运行状况检查行为。除非您手动配置了 Amazon EC2 Auto Scaling 进行替换，否则 Amazon EC2 Auto Scaling 不会自动替换未通过 Elastic Load Balancing 运行状况检查的实例。有关详细信息，请参阅Auto Scaling 运行状况检查设置。

有关运行状况检查以及其对环境的总体运行状况的影响的更多信息，请参阅基本运行状况报告。

使用 EB CLI 配置经典负载均衡器

当您运行 eb create 时，EB CLI 会提示您选择负载均衡器类型。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
```

```
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF

Select a load balancer type
1) classic
2) application
3) network
(default is 1):
```

按 Enter 以选择 `classic`。

您也可以使用 `--elb-type` 选项指定负载均衡器类型。

```
$ eb create test-env --elb-type classic
```

经典负载均衡器配置命名空间

您可以在以下命名空间中找到与经典负载均衡器相关的设置：

- `aws:elb:healthcheck` – 配置负载均衡器运行状况检查的阈值、检查间隔和超时。
- `aws:elasticbeanstalk:application` – 配置运行状态检查 URL。
- `aws:elb:loadbalancer` – 启用跨区域负载均衡 向负载均衡器分配安全组并覆盖 Elastic Beanstalk 创建的默认安全组。此命名空间还包含一些已弃用的选项，这些选项用于配置已由 `aws:elb:listener` 命名空间中的选项替代的标准侦听器和安全侦听器。
- `aws:elb:listener` – 在端口 80 上配置默认侦听器、在端口 443 上配置安全侦听器或在任何端口上配置针对任何协议的其他侦听器。如果您指定 `aws:elb:listener` 作为命名空间，设置适用于端口 80 上的默认监听器。如果您指定了端口 (例如 `aws:elb:listener:443`)，则在该端口上配置监听器。
- `aws:elb:policies` – 配置负载均衡器的其他设置。使用该命名空间中的选项可在任意端口上配置侦听器，修改其他粘性会话设置，以及配置负载均衡器以安全地连接到 Amazon EC2 实例。

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

Example .ebextensions/loadbalancer-terminatehttps.config

以下示例配置文件将在端口 443 上创建一个 HTTPS 侦听器，分配负载均衡器用于终止安全连接的证书，并禁用端口 80 上的默认侦听器。负载均衡器将已解码的请求转发至环境中 HTTP:80 上的 EC2 实例。

```
option_settings:
  aws:elb:listener:443:
    ListenerProtocol: HTTPS
    SSLCertificateId: arn:aws:acm:us-
east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678
    InstancePort: 80
    InstanceProtocol: HTTP
  aws:elb:listener:
    ListenerEnabled: false
```

配置 Application Load Balancer

启用负载平衡后，您的 AWS Elastic Beanstalk 环境将配备 Elastic Load Balancing 负载均衡器，用于在环境中的实例之间分配流量。Elastic Load Balancing 支持多种负载均衡器类型。要了解更多信息，请参阅 Elastic Load Balancing 用户指南。Elastic Beanstalk 可以为您创建负载均衡器，或者让您指定已创建的共享负载均衡器。

本主题介绍 Elastic Beanstalk 创建并专用于您的环境的 Application Load Balancer 的配置。另请参阅 the section called “共享 Application Load Balancer”。有关配置 Elastic Beanstalk 支持的所有负载均衡器类型的信息，请参阅the section called “负载均衡器”。

> **Note**
>
> 您只能在创建环境期间选择您的环境使用的负载均衡器类型。您可以更改设置以管理运行的环境的负载均衡器行为，但不能更改其类型。您也无法从专用负载均衡器切换到共享负载均衡器，反之亦然。

简介

Application Load Balancer 检查应用程序网络协议层中的流量以确定请求的路径，以便将不同路径的请求传送到不同的目标。

当您的环境使用 Application Load Balancer 时，Elastic Beanstalk 默认情况下将其配置为执行与经典负载均衡器相同的功能。默认侦听器接收端口 80 上的 HTTP 请求并将它们分配到环境中的实例。您可以使用证书在端口 443 上添加一个安全的侦听器，以解密 HTTPS 流量、配置运行状况检查行为和将访问日志从负载均衡器推送到 Amazon Simple Storage Service (Amazon S3) 存储桶。

> **Note**
>
> 与经典负载均衡器或 Network Load Balancer 不同，Application Load Balancer 不能有传输层（第 4 层）TCP 或 SSL/TLS 侦听器。它仅支持 HTTP 和 HTTPS 侦听器。此外，它不能使用后端身份验证对负载均衡器和后端实例之间的 HTTPS 连接进行身份验证。

在 Elastic Beanstalk 环境中，您可以使用 Application Load Balancer 将特定路径的流量定向到 Web 服务器实例上的其他进程。使用经典负载均衡器，到一个侦听器的所有流量都将路由到后端实例上的一个端口。使用 Application Load Balancer，您可以在侦听器上配置多个规则，以将发往特定路径的请求路由到其他后端进程。您可以使用每个进程侦听的端口配置进程。

例如，您可能运行与主应用程序分开的登录进程。您的环境的实例上的主应用程序接受大部分请求并侦听端口 80，而您的登录进程侦听端口 5000 并接受发送到 `/login` 路径的请求。来自客户端的所有传入请求都是在端口 80 上传送的。通过使用 Application Load Balancer，您可以为端口 80 上的传入流量配置单个侦听器，并使用两个规则将流量路由到两个单独的进程，具体取决于请求中的路径。您可以添加自定义规则，将传输到 `/login` 的流量路由到侦听 5000 端口的登录进程。默认规则将所有其他流量路由到侦听端口 80 的主应用程序进程。

Application Load Balancer 规则将请求映射到目标组。在 Elastic Beanstalk 中，目标组是由进程 表示的。您可以为进程配置协议、端口和运行状况检查设置。进程代表在环境中的实例上运行的进程。默认进程是在您的应用程序前面运行的反向代理 (nginx 或 Apache) 的端口 80 上的一个侦听器。

> **Note**
>
> 在 Elastic Beanstalk 外部，一个目标组将映射到一组实例。一个侦听器可使用多个规则和多个目标组来基于路径将流量路由到不同的实例。在 Elastic Beanstalk 内部，您的环境中的所有实例均相同，因此会对在不同端口上侦听的过程进行区分。

经典负载均衡器将单个运行状况检查路径用于整个环境。使用 Application Load Balancer 时，每个进程都有一个单独的运行状况检查路径，由负载均衡器和 Elastic Beanstalk 增强型运行状况监控来监控。

要使用 Application Load Balancer，您的环境必须在默认或自定义 VPC 中，且必须有一个具有标准权限集的服务角色。如果您有较旧的服务角色，您可能需要对其更新权限以包含 `elasticloadbalancing:DescribeTargetHealth` 和 `elasticloadbalancing:DescribeLoadBalancers`。有关 Application Load Balancer 的更多信息，请参阅什么是 Application Load Balancer？

> **Note**
>
> Application Load Balancer 运行状况检查不使用 Elastic Beanstalk 运行状况检查路径。相反，它使用为每个进程单独配置的特定路径。

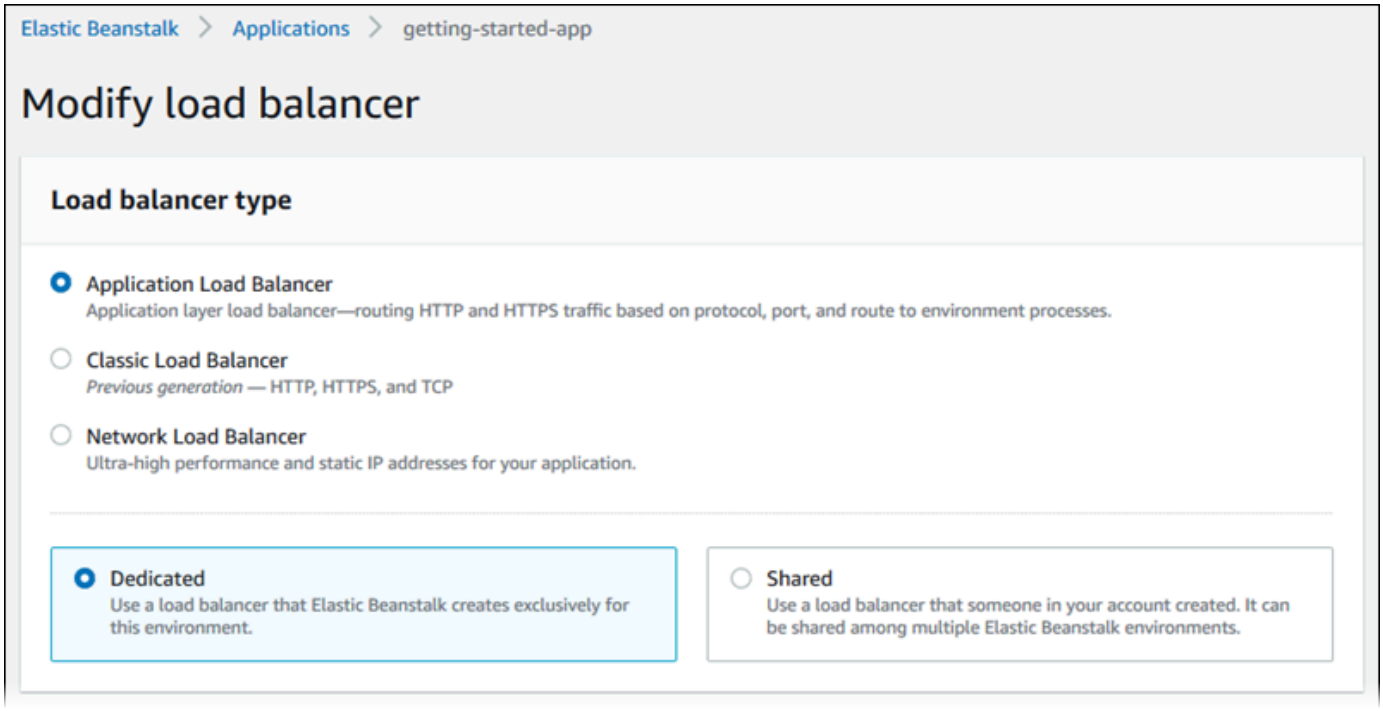
使用 Elastic Beanstalk 控制台配置 Application Load Balancer

在创建环境期间或以后运行您的环境时，您可以使用 Elastic Beanstalk 控制台配置 Application Load Balancer 的侦听器、进程和规则。

环境创建期间，在 Elastic Beanstalk 控制台中配置 Application Load Balancer

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择环境。
3. 选择 Create a new environment (创建新环境) 以开始创建环境。
4. 在向导的主页上，在选择创建环境之前，选择配置更多选项。
5. 选择高可用性配置预设。

   或者，在容量配置类别中配置负载均衡环境类型。有关更多信息，请参阅 容量。
6. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。
7. 如果尚未选择 Application Load Balancer 和 Dedicated (专用) 选项，请选择它们。



8. 根据环境需要进行任何 Application Load Balancer 配置更改。
9. 选择保存，然后进行您的环境所需的任何其他配置更改。
10. 选择创建环境。

在 Elastic Beanstalk 控制台中配置正在运行环境的 Application Load Balancer

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。

> **Note**
>
> 如果 Load balancer (负载均衡器) 配置类别没有 Edit (编辑) 按钮，则表示您的环境没有负载均衡器。要了解如何设置负载均衡器，请参阅更改环境类型。

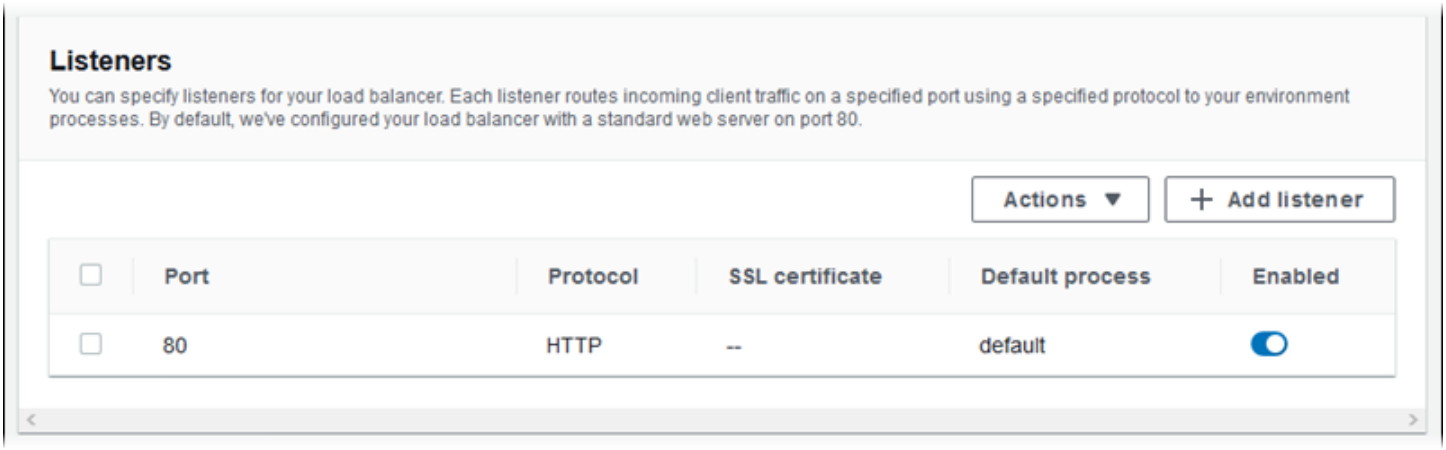
5. 根据环境需要进行 Application Load Balancer 配置更改。
6. 要保存更改，请选择页面底部的 Apply（应用）。

Application Load Balancer 设置

- 侦听器
- 进程
- 规则
- 访问日志捕获

侦听器

可以使用该列表为您的负载均衡器指定侦听器。每个侦听器使用指定协议将在指定端口上传入的客户端流量路由到实例上的一个或多个进程。最初，该列表显示默认侦听器，它将端口 80 上的传入 HTTP 流量路由到名为 default 的进程。



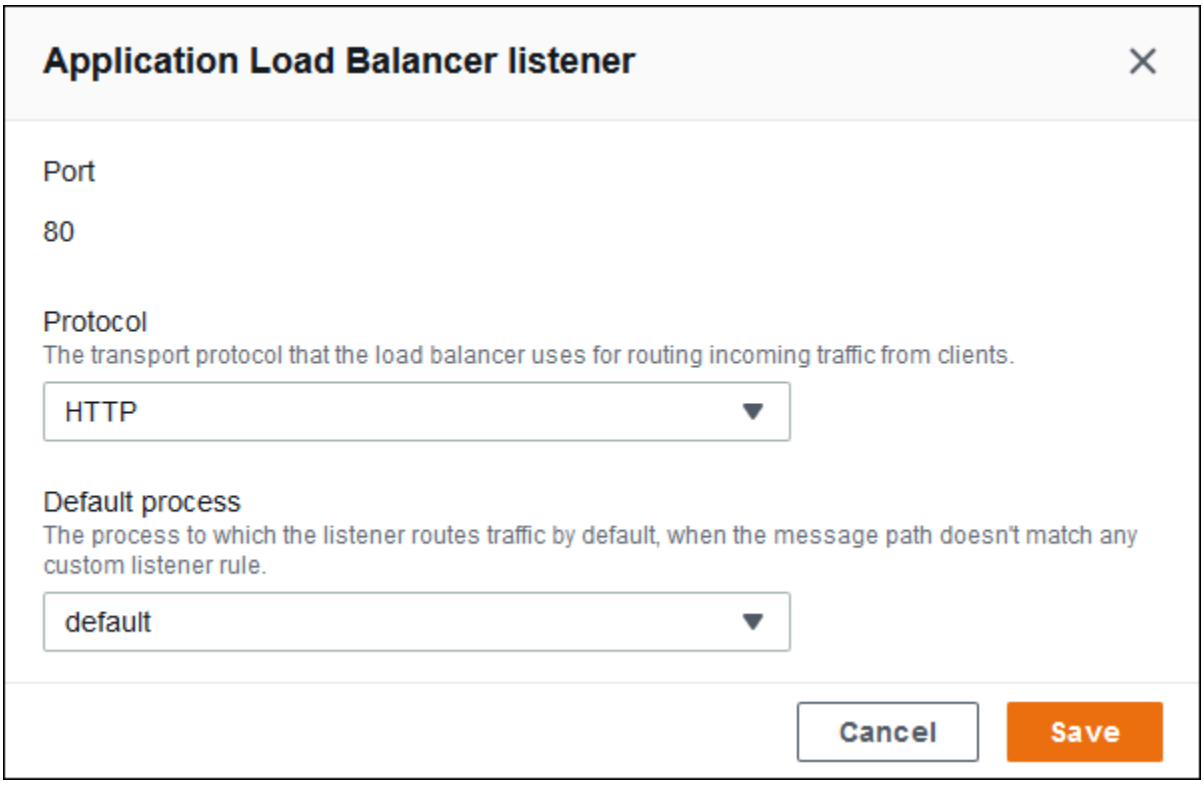
配置现有的侦听器

1. 选中表条目旁边的复选框，然后选择操作和编辑。
2. 使用 Application Load Balancer 侦听器对话框编辑设置，然后选择保存。

添加侦听器

1. 选择添加侦听器。
2. 在 Application Load Balancer 侦听器对话框中，配置所需的设置，然后选择添加。

使用 Application Load Balancer 侦听器对话框中的设置选择侦听器用于侦听流量的端口和协议，以及要将流量路由到的进程。如果您选择 HTTPS 协议，请配置 SSL 设置。



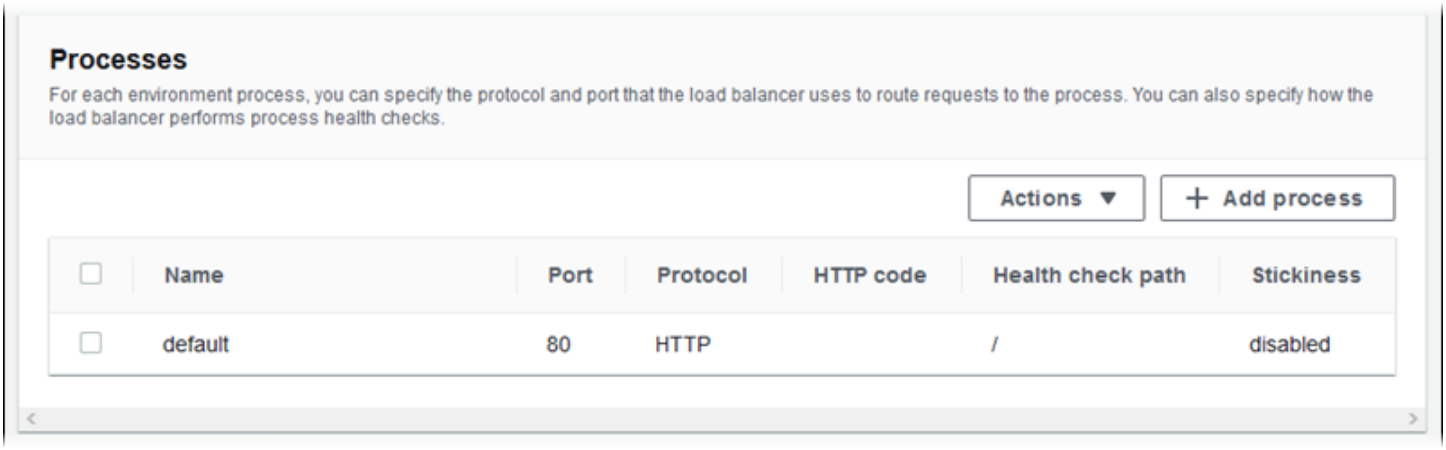
确保您具有有效的 SSL 证书，然后才能配置 HTTPS 侦听器。请执行以下操作之一：

- 如果 AWS Certificate Manager (ACM) 在您所在的 AWS 地区可用，请使用 ACM 创建或导入证书。有关请求 ACM 证书的更多信息，请参阅 AWS Certificate Manager 用户指南中的请求证书。有关将第三方证书导入 ACM 中的更多信息，请参阅 AWS Certificate Manager 用户指南中的导入证书。
- 如果您所在的AWS 地区没有 ACM，请将您现有的证书和密钥上传到 IAM。有关创建证书并将证书上传到 IAM 的更多信息，请参阅《IAM 用户指南》中的使用服务器证书。

有关在 Elastic Beanstalk 中配置 HTTPS 和使用证书的更多详细信息，请参阅为 Elastic Beanstalk 环境配置 HTTPS。

进程

可以使用该列表为您的负载均衡器指定进程。进程是侦听器将流量路由到的目标。每个侦听器使用指定协议将在指定端口上传入的客户端流量路由到实例上的一个或多个进程。最初，该列表显示默认进程，它侦听端口 80 上的传入 HTTP 流量。



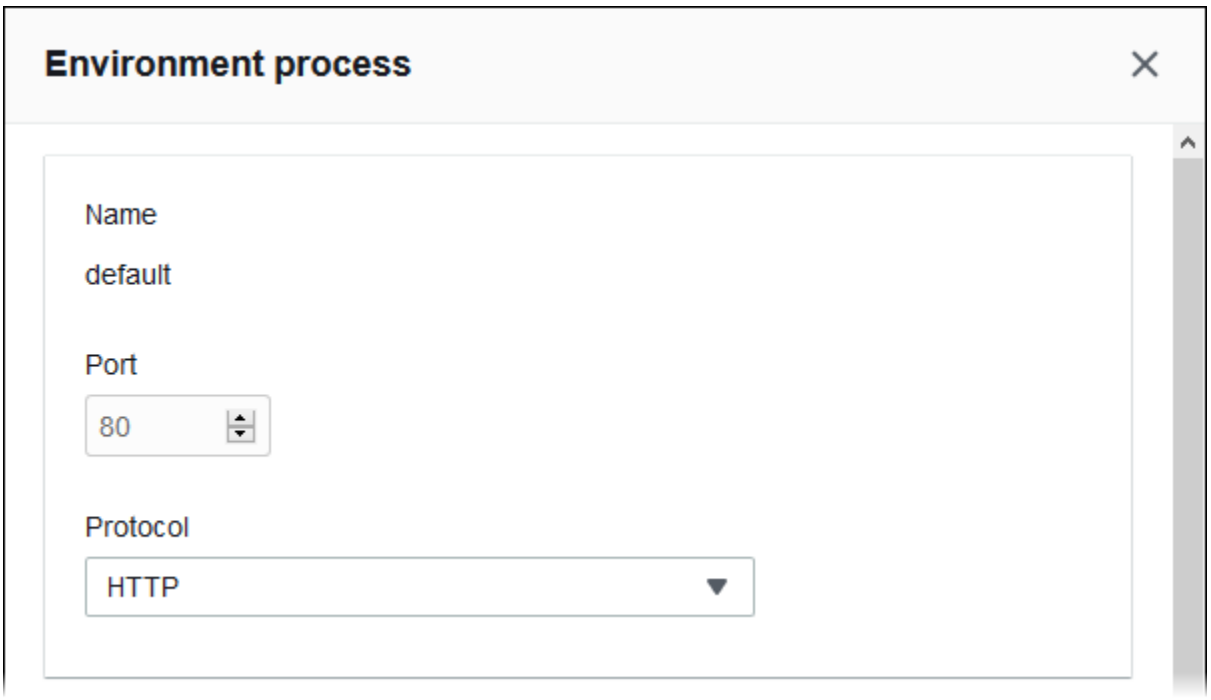
您可以编辑现有进程的设置或添加新的进程。要在列表中开始编辑或添加进程，请使用为侦听器列表列出的相同步骤。将打开环境进程对话框。

Application Load Balancer 的环境进程对话框设置

- 定义
- 运行状况检查
- 会话

定义

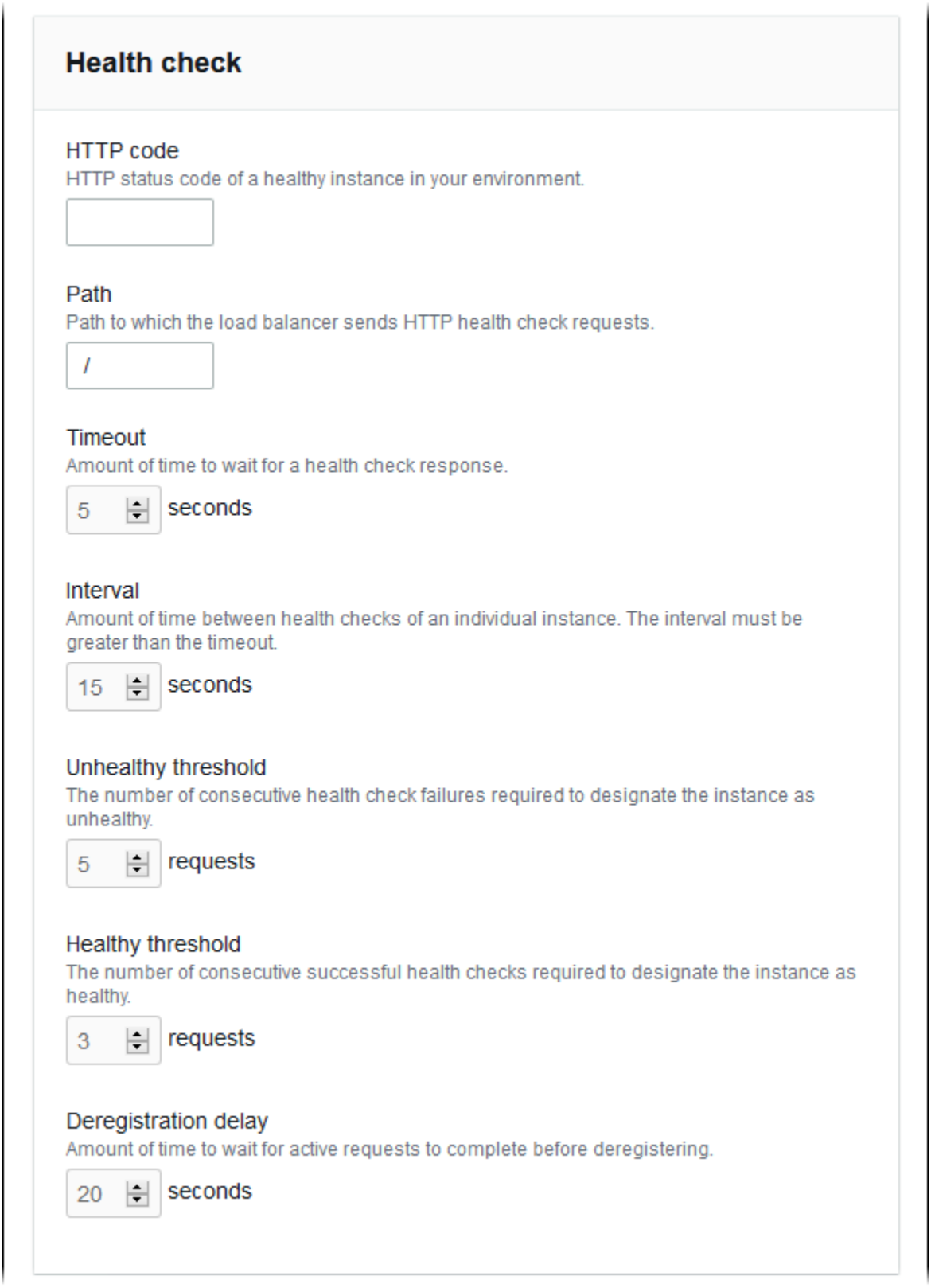
可以使用以下设置定义进程：其名称以及它在其中侦听请求的端口和协议。



运行状况检查

可以使用以下设置配置进程运行状况检查：

- HTTP code (HTTP 代码) – 指定正常运行的进程的 HTTP 状态代码。
- Path (路径) – 进程的运行状况检查请求路径。
- Timeout (超时) – 等待运行状况检查响应的时间（秒）。
- Interval (间隔) – 单个实例的两次运行状况检查间隔的时间（秒）。间隔必须大于超时。
- Unhealthy threshold (不正常阈值) 和 Healthy threshold (正常阈值) – 在 Elastic Load Balancing 更改实例的运行状况之前，实例必须通过或未通过的运行状况检查次数。
- Deregistration delay (取消注册延迟) – 在取消注册实例之前等待活动请求完成的时间（秒）。

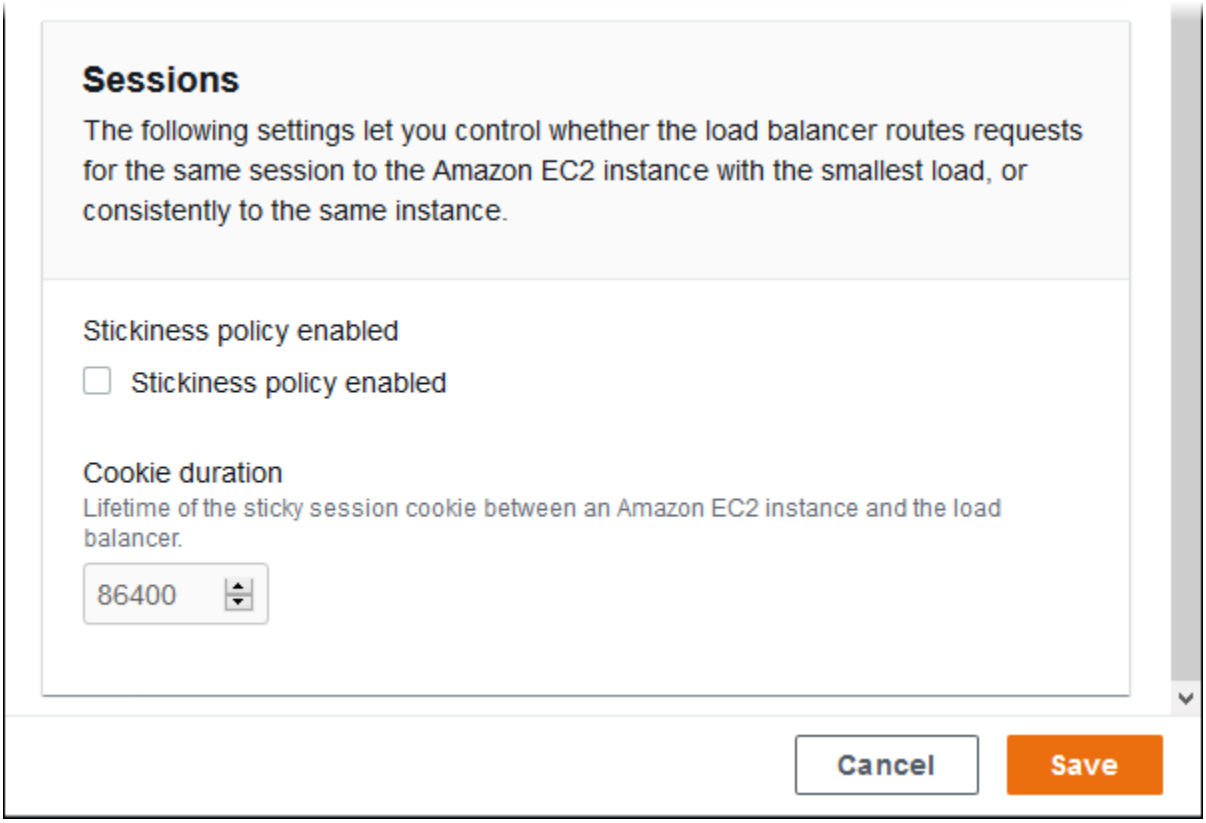

Health check
HTTP code
HTTP status code of a healthy instance in your environment.
Path
Path to which the load balancer sends HTTP health check requests.
/
Timeout
Amount of time to wait for a health check response.
5 seconds
Interval
Amount of time between health checks of an individual instance. The interval must be greater than the timeout.
15 seconds
Unhealthy threshold
The number of consecutive health check failures required to designate the instance as unhealthy.
5 requests
Healthy threshold
The number of consecutive successful health checks required to designate the instance as healthy.
3 requests
Deregistration delay
Amount of time to wait for active requests to complete before deregistering.
20 seconds

> **Note**
>
> Elastic Load Balancing 运行状况检查不会影响环境的 Auto Scaling 组的运行状况检查行为。除非您手动配置了 Amazon EC2 Auto Scaling 进行替换，否则 Amazon EC2 Auto Scaling 不会自动替换未通过 Elastic Load Balancing 运行状况检查的实例。有关详细信息，请参阅 Auto Scaling 运行状况检查设置。

有关运行状况检查以及其对环境的总体运行状况的影响的更多信息，请参阅基本运行状况报告。

会话

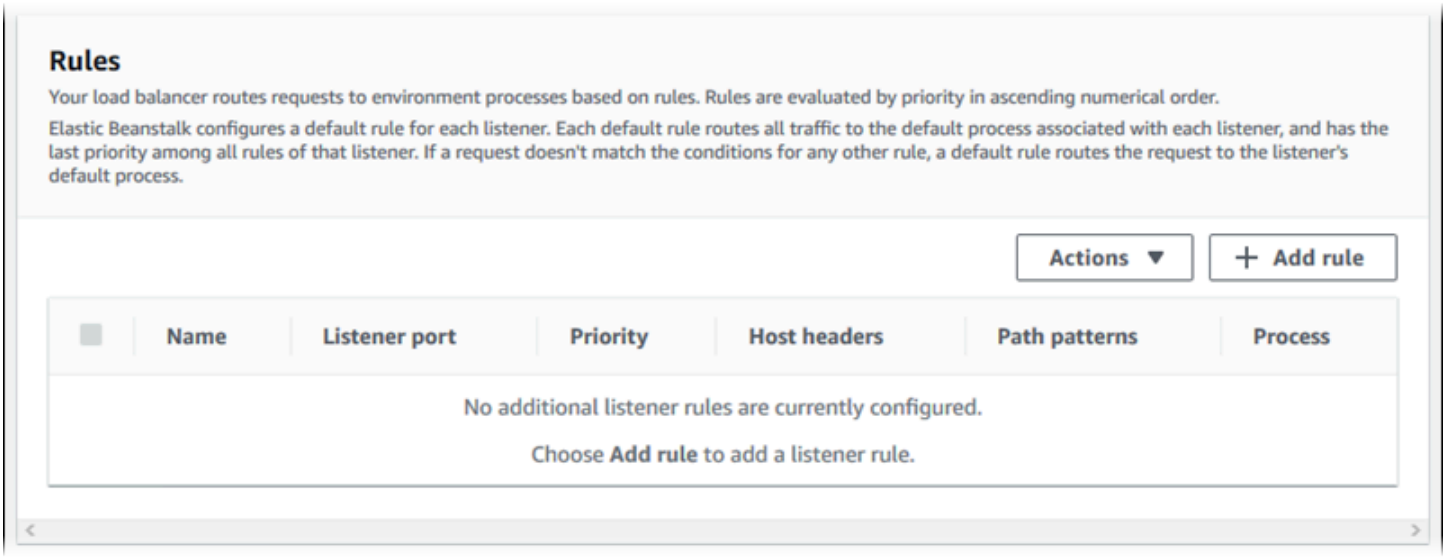
选中或清除粘性策略已启用框来启用或禁用粘性会话。使用 Cookie 持续时间配置粘性会话的持续时间，最多为 **604800** 秒。



规则

可以使用该列表为负载均衡器指定自定义侦听器规则。规则将侦听器在特定路径模式上接收的请求映射到目标进程。每个侦听器可以具有多个规则，以将不同路径上的请求路由到实例上的不同进程。

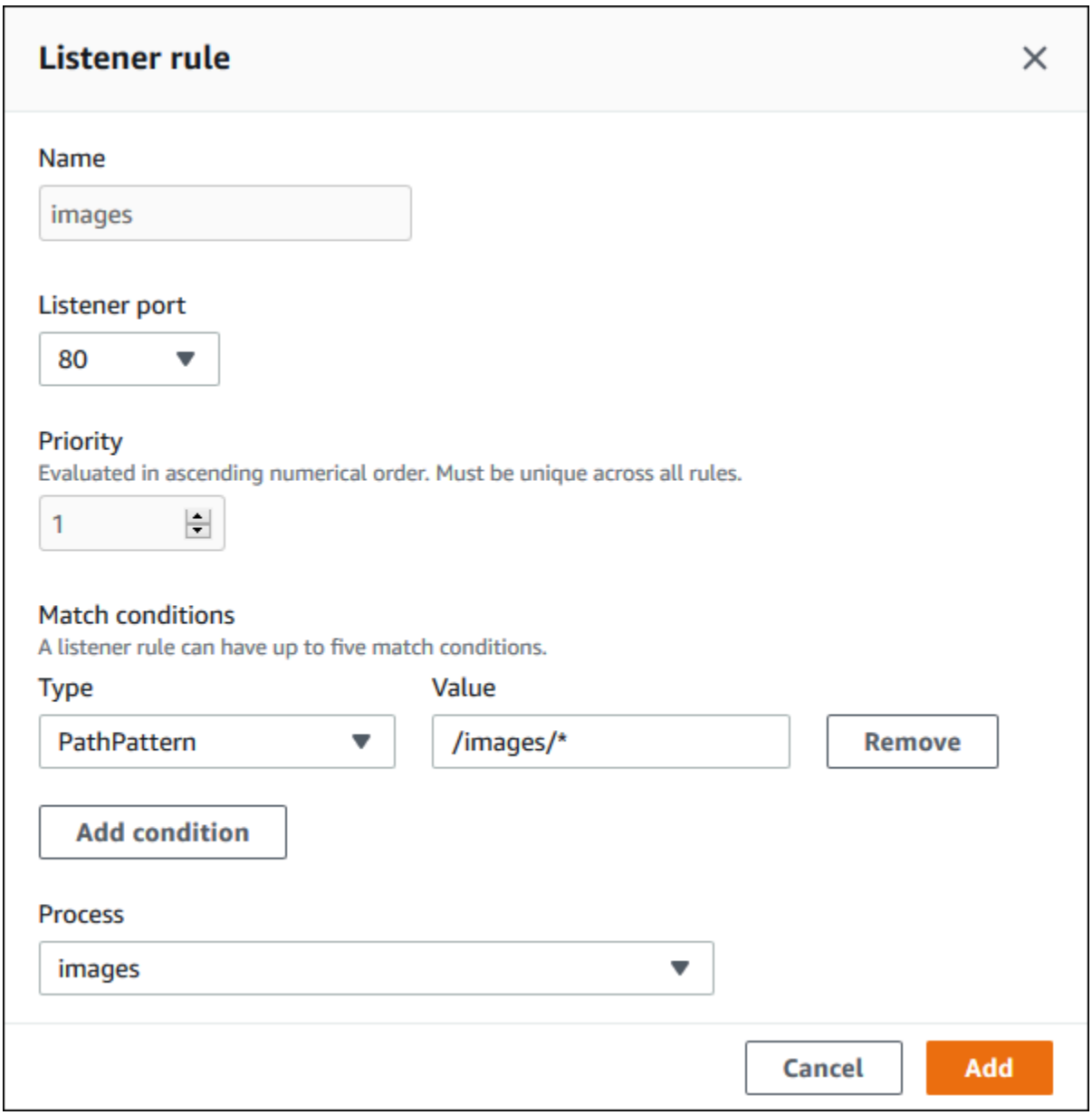
规则具有数字优先级，它确定将规则应用于传入请求的优先顺序。对于您添加的每个新侦听器，Elastic Beanstalk 添加一个默认规则，它将侦听器的所有流量路由到默认进程。默认规则的优先顺序最低；如果同一侦听器没有与传入请求匹配的任何其他规则，则应用该规则。起初在您还未添加自定义规则时，该列表是空的。不会显示所有侦听器的默认规则。



您可以编辑现有规则的设置或添加新的规则。要在列表中开始编辑或添加规则，请使用为侦听器列表列出的相同步骤。将打开侦听器规则对话框并包含以下设置：

- Name (名称) – 规则的名称。
- Listener port (侦听器端口) – 规则应用到的侦听器端口。
- Priority (优先级) – 规则的优先级。优先级序号越小，优先顺序越靠前。侦听器的规则优先级必须是唯一的。
- Match conditions (匹配条件) – 规则应用到的请求 URL 条件的列表。有两种类型的条件：HostHeader（网址的网域部分）和 PathPattern（网址的路径部分）。您最多可以添加五个条件。每个条件值的长度最多为 128 个字符，并且可以包含通配符。
- Process (进程) – 负载均衡器将与规则匹配的请求路由到的进程。

在编辑任何现有规则时，您无法更改其名称和侦听器端口。

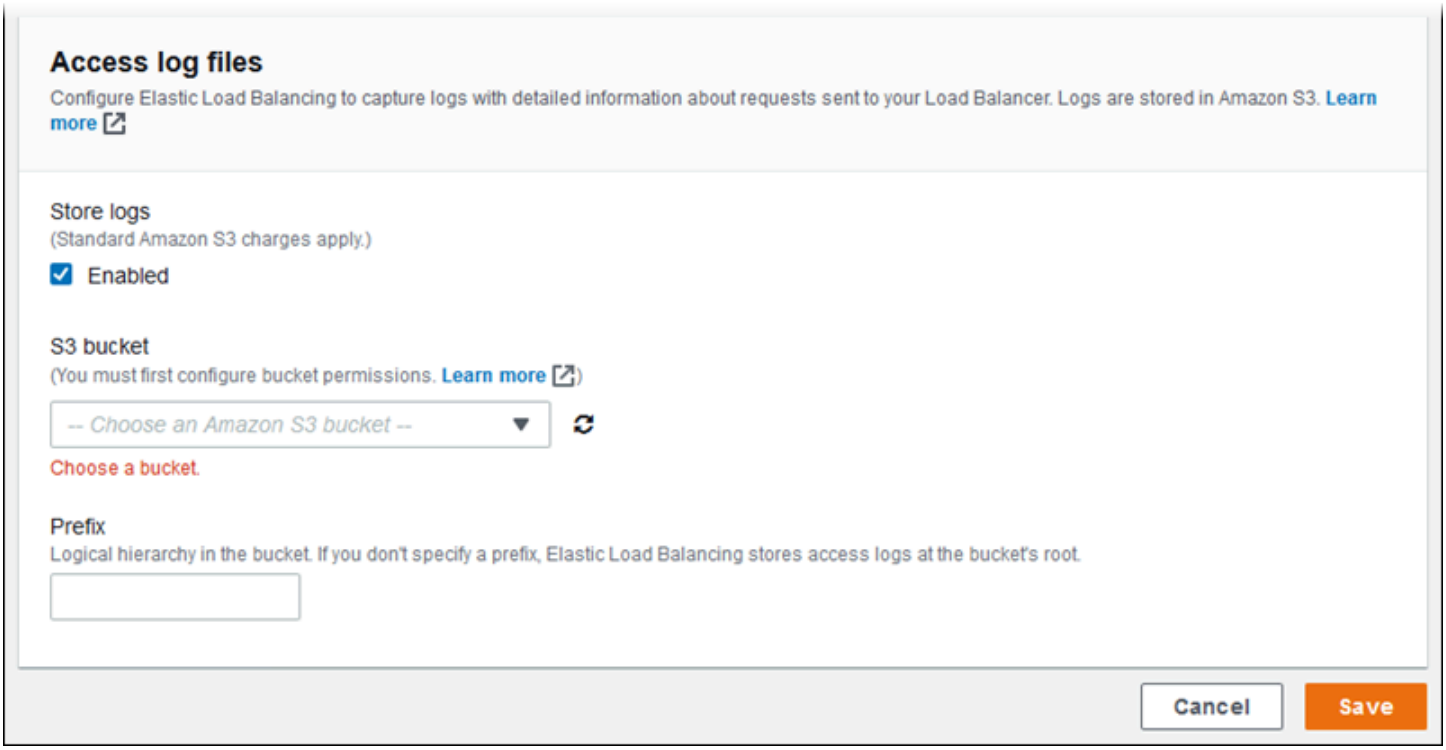


访问日志捕获

使用这些设置配置 Elastic Load Balancing ，使其捕获包含有关发送到 Application Load Balancer 的请求详细信息的日志。默认情况下，已禁用访问日志捕获。在启用 Store logs (存储日志) 时，Elastic Load Balancing 将日志存储在您配置的 S3 bucket (S3 存储桶) 中。Prefix (前缀) 设置为日志指定存储桶中的顶级文件夹。Elastic Load Balancing 将日志放置在您前缀下名为 `AWSLogs` 的文件夹中。如果您不指定前缀，则 Elastic Load Balancing 会将其文件夹置于存储桶的根级。

> **Note**
>
> 如果您为访问日志捕获配置的 Amazon S3 存储桶不是 Elastic Beanstalk 为您的账户创建的存储桶，请务必为您的 (IAM) 用户添加具有相应权限 AWS Identity and Access Management 的用户策略。Elastic Beanstalk 提供的托管用户策略仅涵盖对 Elastic Beanstalk 托管资源的权限。

有关访问日志的详细信息，包括权限和其他要求，请参阅 Application Load Balancer 的访问日志。



示例：具有安全侦听器和两个进程的 Application Load Balancer

在此示例中，您的应用程序需要 end-to-end 流量加密和单独的处理管理请求的流程。

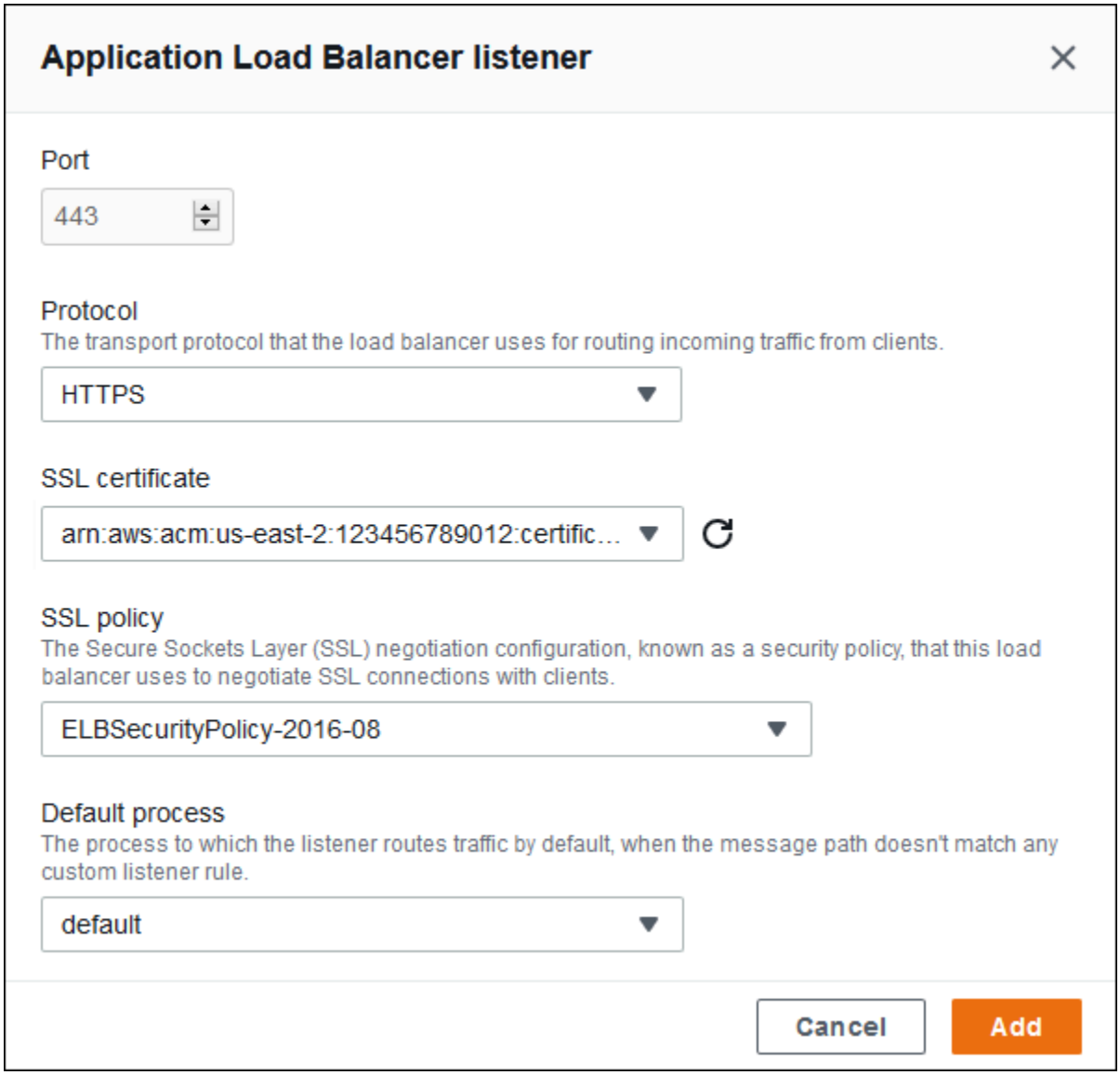
为了配置您的环境的 Application Load Balancer 以满足这些要求，您删除默认侦听器，添加一个 HTTPS 侦听器，指示默认进程侦听端口 443 的 HTTPS，并为不同路径上的管理流量添加一个进程和侦听器规则。

为该示例配置负载均衡器

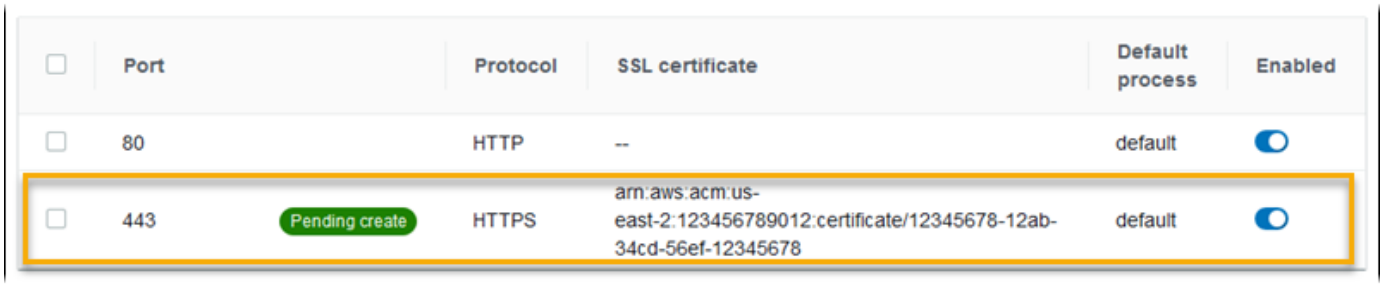
1. 添加安全侦听器。对于端口，输入 **443**。对于协议，选择 **HTTPS**。对于 SSL 证书，选择 SSL 证书的 ARN。例如，**arn:aws:iam::123456789012:server-**

certificate/abc/certs/build 或 **arn:aws:acm:us-east-2:123456789012:certificate/12345678-12ab-34cd-56ef-12345678**。

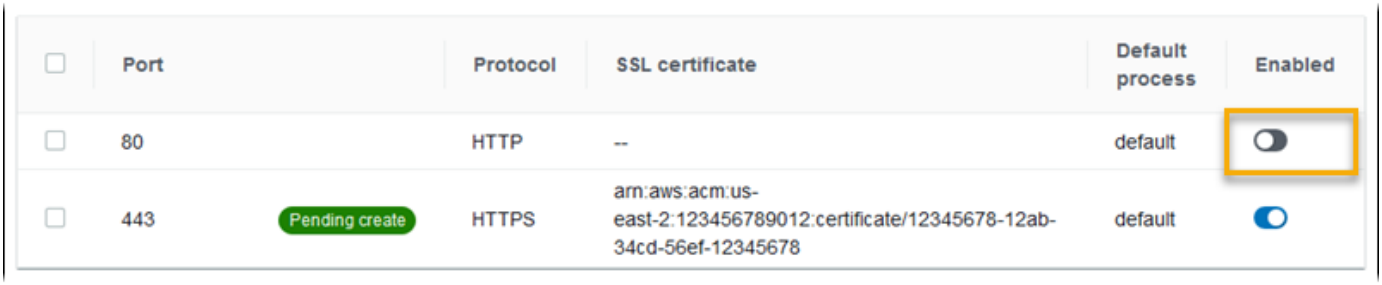
对于默认进程，保持选中 **default**。



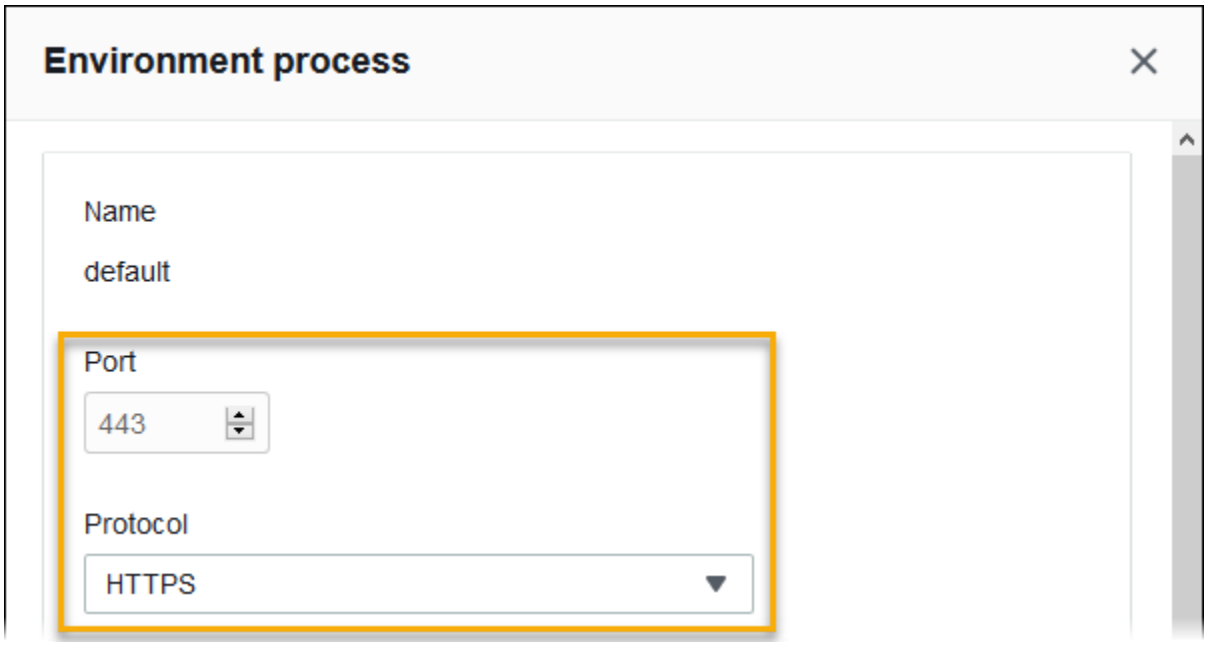
您现在可以查看列表上的其他侦听器。



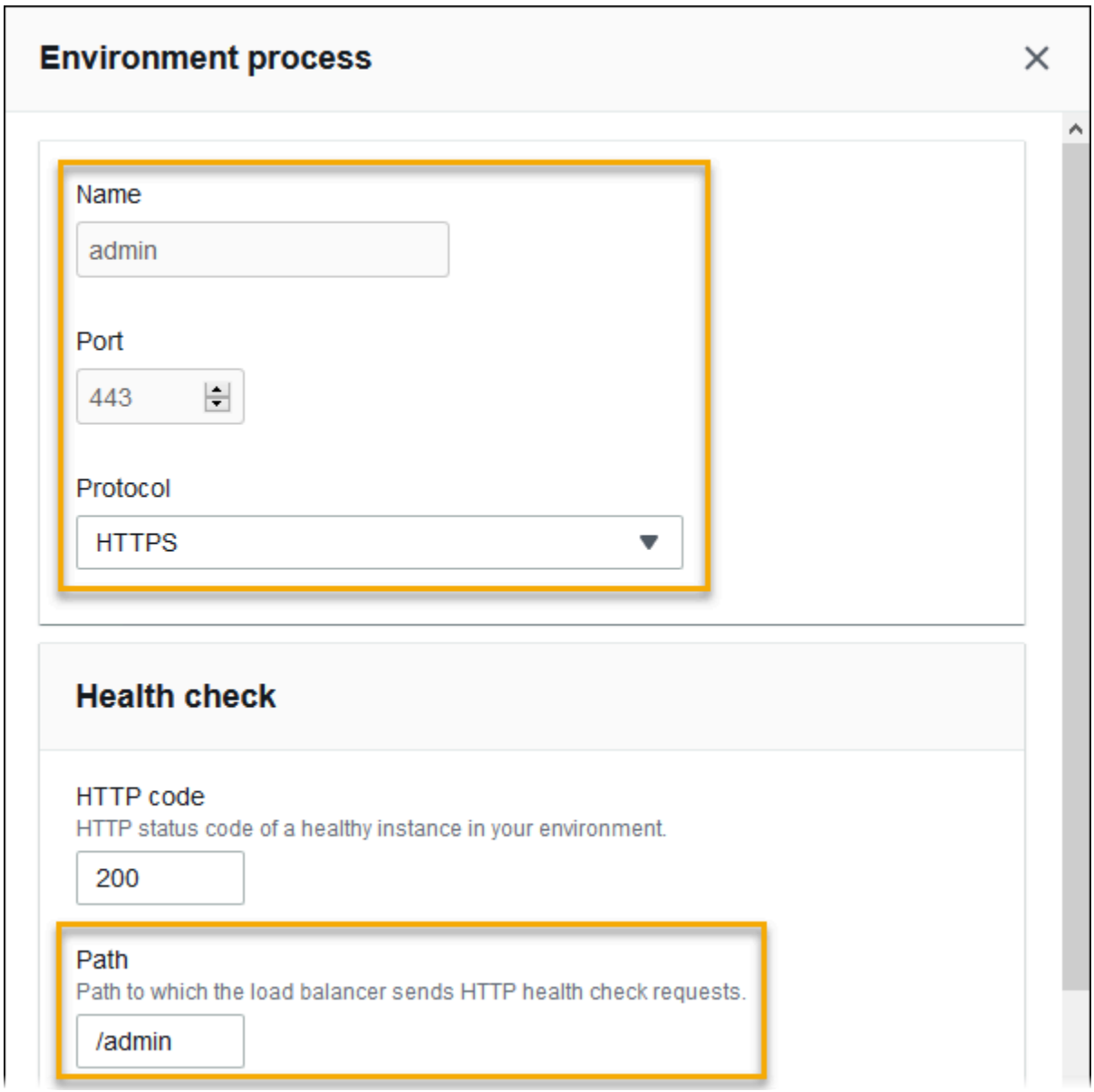
2. 禁用默认端口 80 HTTP 侦听器。对于默认侦听器，关闭已启用选项。



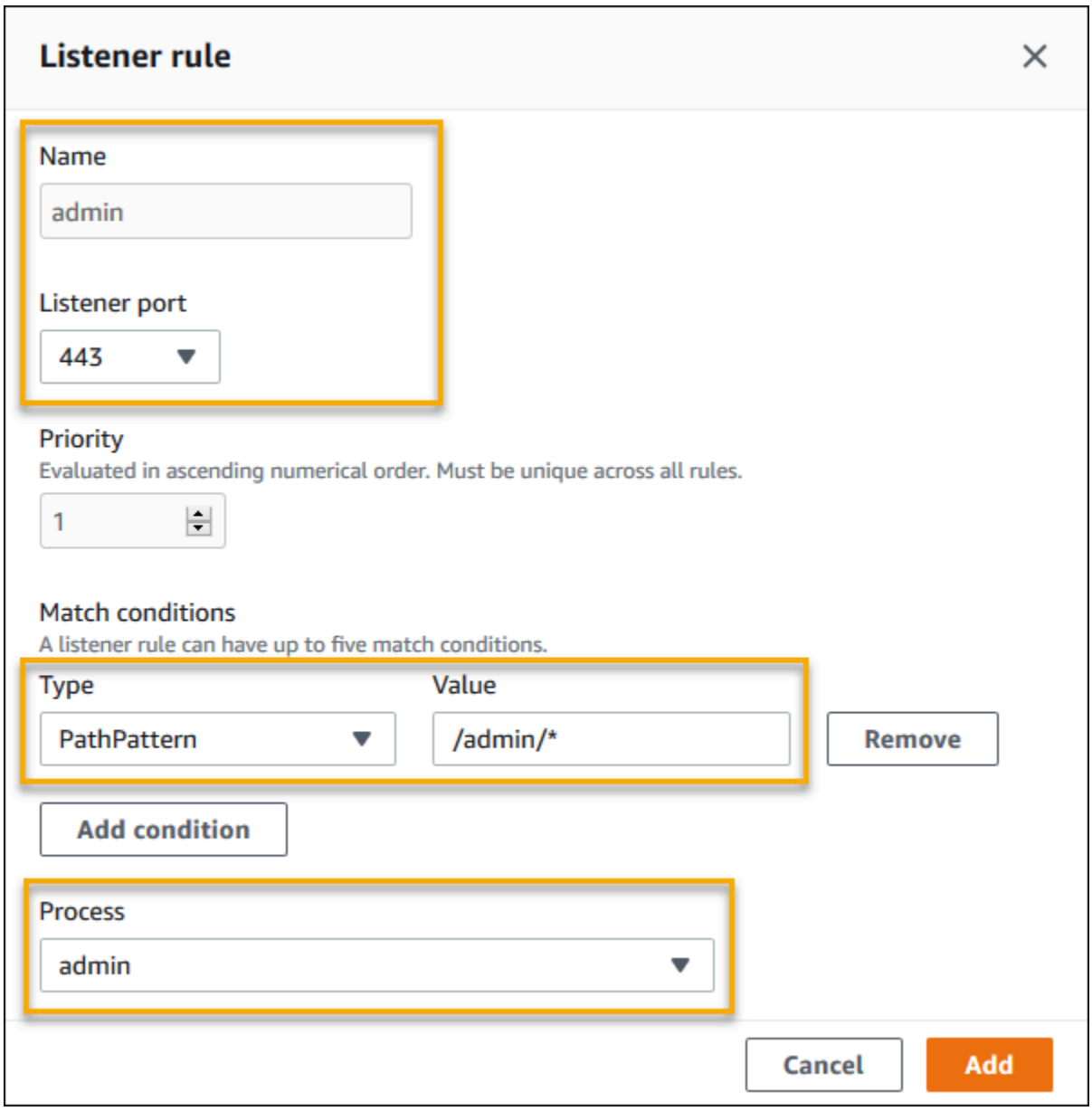
3. 将默认进程配置为 HTTPS。选择默认进程，然后为操作选择编辑。对于端口，输入 **443**。对于协议，选择 **HTTPS**。



4. 添加一个管理进程。对于名称，键入 **admin**。对于端口，输入 **443**。对于协议，选择 **HTTPS**。在运行状况检查下，为路径输入 **/admin**。



5. 为管理流量添加一个规则。对于名称，键入 **admin**。对于 Listener port (侦听器端口)，键入 **443**。在 “匹配条件” 中，添加值为 a PathPattern**/admin/***。对于进程，选择 **admin**。



使用 EB CLI 配置 Application Load Balancer

当您运行 eb create 时，EB CLI 会提示您选择负载均衡器类型。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF

Select a load balancer type
```

```
1) classic
2) application
3) network
(default is 2):
```

您也可以使用 `--elb-type` 选项指定负载均衡器类型。

```
$ eb create test-env --elb-type application
```

Application Load Balancer 命名空间

您可以在以下命名空间中找到与 Application Load Balancer 相关的设置：

- `aws:elasticbeanstalk:environment` – 选择用于环境的负载均衡器类型。Application Load Balancer 的值为 `application`。

  无法在配置文件 (.Ebextensions) 中设置此选项。
- `aws:elbv2:loadbalancer` – 将访问日志和应用于 Application Load Balancer 的其他设置作为一个整体配置。
- `aws:elbv2:listener` – 在 Application Load Balancer 上配置侦听器。这些设置将映射到经典负载均衡器的 `aws:elb:listener` 中的设置。
- `aws:elbv2:listenerrule` – 配置根据请求路径将流量路由到不同过程的规则。规则对于 Application Load Balancers 唯一。
- `aws:elasticbeanstalk:environment:process` – 配置运行状况检查并为在您的环境的实例上运行的过程指定端口和协议。端口和协议设置将映射到经典负载均衡器上侦听器的 `aws:elb:listener` 中的实例端口和实例协议设置。运行状况检查设置将映射到 `aws:elb:healthcheck` 和 `aws:elasticbeanstalk:application` 命名空间中的设置。

Example .ebextensions/ .config alb-access-logs

以下配置文件为带有 Application Load Balancer 的环境启用访问日志上传：

```
option_settings:
  aws:elbv2:loadbalancer:
    AccessLogsS3Bucket: DOC-EXAMPLE-BUCKET
    AccessLogsS3Enabled: 'true'
    AccessLogsS3Prefix: beanstalk-alb
```

Example .ebextensions/ .config alb-default-process

以下配置文件将修改默认进程的运行状况检查设置和粘性设置。

```
option_settings:
  aws:elasticbeanstalk:environment:process:default:
    DeregistrationDelay: '20'
    HealthCheckInterval: '15'
    HealthCheckPath: /
    HealthCheckTimeout: '5'
    HealthyThresholdCount: '3'
    UnhealthyThresholdCount: '5'
    Port: '80'
    Protocol: HTTP
    StickinessEnabled: 'true'
    StickinessLBCookieDuration: '43200'
```

Example .ebextensions/ .config alb-secure-listener

以下配置文件将在端口 443 上添加一个安全侦听器和匹配进程。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
    Protocol: HTTPS
    SSLCertificateArns: arn:aws:acm:us-
east-2:123456789012:certificate/21324896-0fa4-412b-bf6f-f362d6eb6dd7
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
    Protocol: HTTPS
```

Example .ebextensions/ .config alb-admin-rule

以下配置文件将添加一个安全侦听器，该侦听器具有一个规则，它使用请求路径 `/admin` 将流量路由到名为 `admin` 的在端口 4443 上进行侦听的进程。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
    Protocol: HTTPS
    Rules: admin
```

```
    SSLCertificateArns: arn:aws:acm:us-
east-2:123456789012:certificate/21324896-0fa4-412b-bf6f-f362d6eb6dd7
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
    Protocol: HTTPS
  aws:elasticbeanstalk:environment:process:admin:
    HealthCheckPath: /admin
    Port: '4443'
    Protocol: HTTPS
  aws:elbv2:listenerrule:admin:
    PathPatterns: /admin/*
    Priority: 1
    Process: admin
```

配置共享 Application Load Balancer

在启用负载均衡时，将为您的 AWS Elastic Beanstalk 环境配备 Elastic Load Balancing 负载均衡器，以便在您的环境中的实例之间分配流量。Elastic Load Balancing 支持多种负载均衡器类型。要了解更多信息，请参阅 Elastic Load Balancing 用户指南。Elastic Beanstalk 可以为您创建负载均衡器，或让您指定您已创建的共享负载均衡器。

本主题介绍您创建的并与环境关联的共享 Application Load Balancer 配置。另请参阅 the section called “应用程序负载均衡器”。有关配置 Elastic Beanstalk 支持的所有负载均衡器类型的信息，请参阅Elastic Beanstalk 环境的负载均衡器。

> **Note**
>
> 您只能在创建环境期间选择您的环境使用的负载均衡器类型。您可以更改设置以管理运行的环境的负载均衡器行为，但不能更改其类型。您也无法从专用负载均衡器切换到共享负载均衡器，反之亦然。

介绍

您可以借助 Amazon Elastic Compute Cloud (Amazon EC2) 服务自行创建和管理共享负载均衡器，然后在多个 Elastic Beanstalk 环境中使用。

在您创建负载均衡的自动扩展环境并选择使用 Application Load Balancer 时，Elastic Beanstalk 默认情况下都会相应创建一个专用于您的环境的负载均衡器。要了解什么是 Application Load Balancer 以及它如何用于 Elastic Beanstalk 环境，请参阅为 Elastic Beanstalk 配置 Application Load Balancer 的简介。

在某些情况下，您可能想要节省拥有多个专用负载均衡器的成本。当您拥有多个环境时，这可能会很有帮助，例如，当您的应用程序是一组微服务，而不是一个整体服务时。在这种情况下，您可以选择使用共享的负载均衡器。

要使用共享的负载均衡器，请先在 Amazon EC2 中创建该负载均衡器，然后添加一个或多个侦听器。在创建 Elastic Beanstalk 环境期间，您随之提供负载均衡器并选择侦听器端口。Elastic Beanstalk 将侦听器与您环境中的默认进程相关联。您可以添加自定义侦听器规则，以将流量从特定主机标头和路径路由到其他环境进程。

Elastic Beanstalk 将标签添加到共享负载均衡器。标签名称为 `elasticbeanstalk:shared-elb-environment-count`，其值是共享此负载均衡器的环境数。

使用共享负载均衡器与使用专用负载均衡器在多个方面有所不同。

关于	专用 Application Load Balancer	共享 Application Load Balancer
管理	Elastic Beanstalk 创建和管理负载均衡器、侦听器、侦听器规则和进程（目标组）。Elastic Beanstalk 还会在您终止环境时删除它们。如果选择该选项，Elastic Beanstalk 可以设置负载均衡器访问日志捕获。	您可以在 Elastic Beanstalk 之外创建并管理负载均衡器和侦听器。Elastic Beanstalk 创建并管理默认规则和默认进程，您可以添加规则和进程。Elastic Beanstalk 将删除环境创建期间添加的侦听器规则和进程。
侦听器规则	Elastic Beanstalk 为每个侦听器创建一个默认规则，以将所有流量路由到侦听器的默认进程。	Elastic Beanstalk 仅将默认规则与端口 80 侦听器关联（如果存在）。如果选择不同的默认侦听器端口，则必须将默认规则与其关联（Elastic Beanstalk 控制台和 EB CLI 为您执行此操作）。 为解决共享负载均衡器的各环境间的侦听器规则条件冲突，Elastic Beanstalk 将环境的 CNAME 作为主机标头条件添加到侦听器规则中。 在共享负载均衡器的环境间，Elastic Beanstalk 将规则优先级设置视为相对优先级，并在创建过程中将其映射到绝对优先级。

关于	专用 Application Load Balancer	共享 Application Load Balancer
安全组	Elastic Beanstalk 创建默认安全组并将其附加到负载均衡器。	您可以配置一个或多个安全组以用于负载均衡器。如果您未配置，Elastic Beanstalk 将检查 Elastic Beanstalk 管理的某个现有安全组是否已连接到负载均衡器。如果没有，Elastic Beanstalk 将创建一个安全组并将其附加到负载均衡器。当最后一个共享负载均衡器的环境终止时，Elastic Beanstalk 将删除此安全组。
更新	您可以在创建环境后更新您的 Application Load Balancer。您可以编辑侦听器、侦听器规则和进程。您可以配置负载均衡器访问日志捕获。	您不能使用 Elastic Beanstalk 在 Application Load Balancer 中配置访问日志捕获，也无法在创建环境后更新侦听器和侦听器规则。您只能更新进程（目标组）。要配置访问日志捕获以及更新侦听器和侦听器规则，请使用 Amazon EC2。

使用 Elastic Beanstalk 控制台配置共享 Application Load Balancer

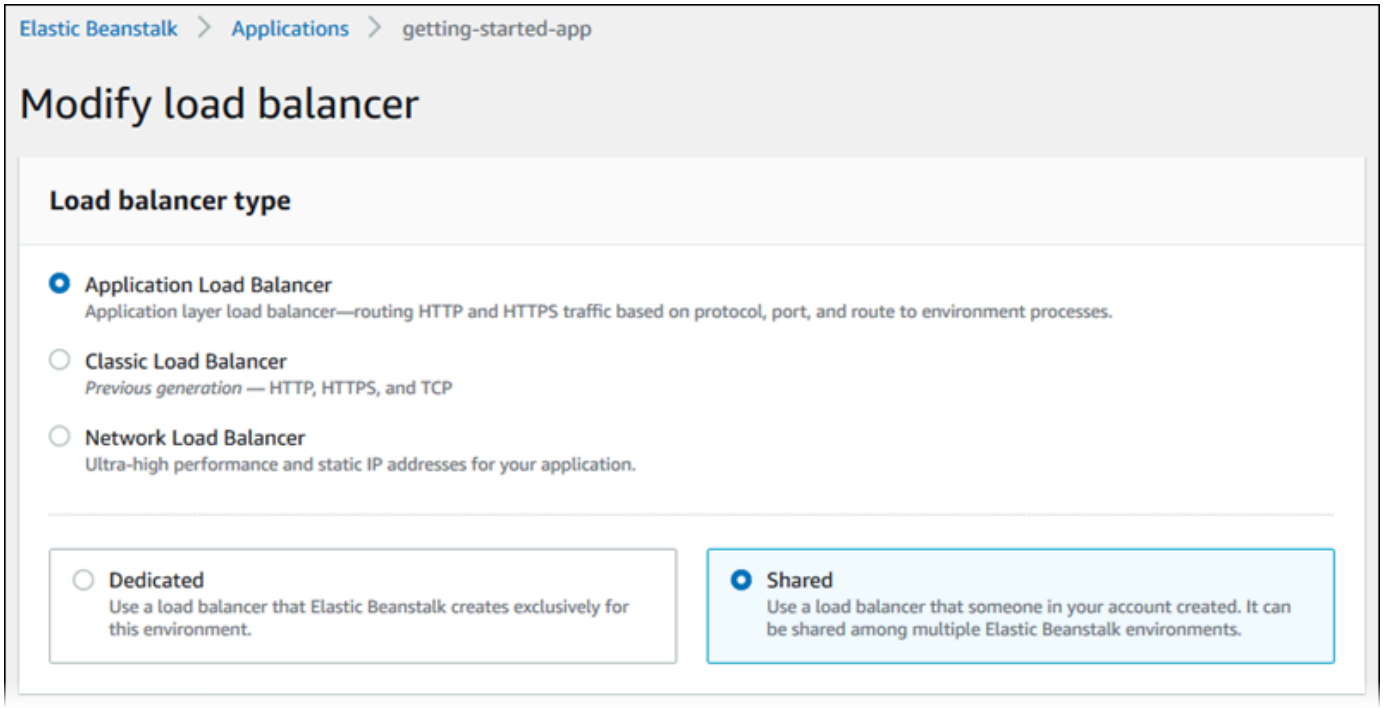
环境创建期间，您可以使用 Elastic Beanstalk 控制台配置共享 Application Load Balancer。您可以选择账户的其中一个可共享的负载均衡器以供在环境中使用，选择默认侦听器端口，并配置其他进程和侦听器规则。

创建环境后，您无法在 Application Load Balancer 控制台中编辑共享 Application Load Balancer 配置。要配置侦听器、侦听器规则、进程（目标组）和访问日志捕获，请使用 Amazon EC2。

环境创建期间，在 Elastic Beanstalk 控制台中配置 Application Load Balancer

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments (环境)。
3. 选择 Create a new environment (创建新环境) 以开始创建环境。
4. 在向导的主页上，在选择创建环境之前，选择配置更多选项。
5. 选择高可用性配置预设。

   或者，在容量配置类别中配置负载均衡环境类型。有关详细信息，请参阅 容量。
6. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。
7. 如果尚未选择该选项，请选择 Application Load Balancer 选项，然后选择 Shared (已共享) 选项。



8. 根据环境需要进行任何共享 Application Load Balancer 配置更改。
9. 选择保存，然后进行您的环境所需的任何其他配置更改。
10. 选择 Create environment（创建环境）。

共享 Application Load Balancer 设置

- 共享 Application Load Balancer
- 进程
- 规则

共享 Application Load Balancer

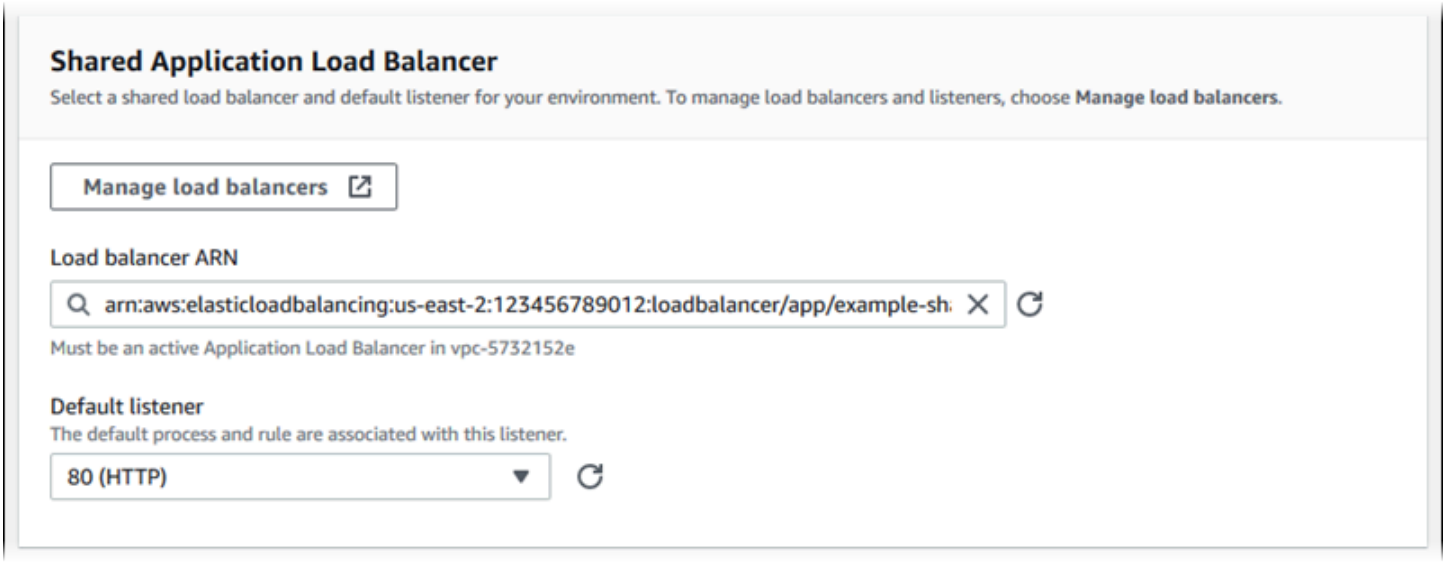
使用此部分可以为您的环境选择共享的 Application Load Balancer 并配置默认流量路由。

在此处配置共享 Application Load Balancer 之前，请使用 Amazon EC2 定义至少一个应用程序负载均衡器，以便与您账户中的至少一个侦听器共享。如果您尚未执行此操作，则可以选择 Manage load balancers (管理负载均衡器)。Elastic Beanstalk 将在新的浏览器标签页中打开 Amazon EC2 控制台。

在 Elastic Beanstalk 外部配置完共享的负载均衡器后，请在此控制台部分配置以下设置：

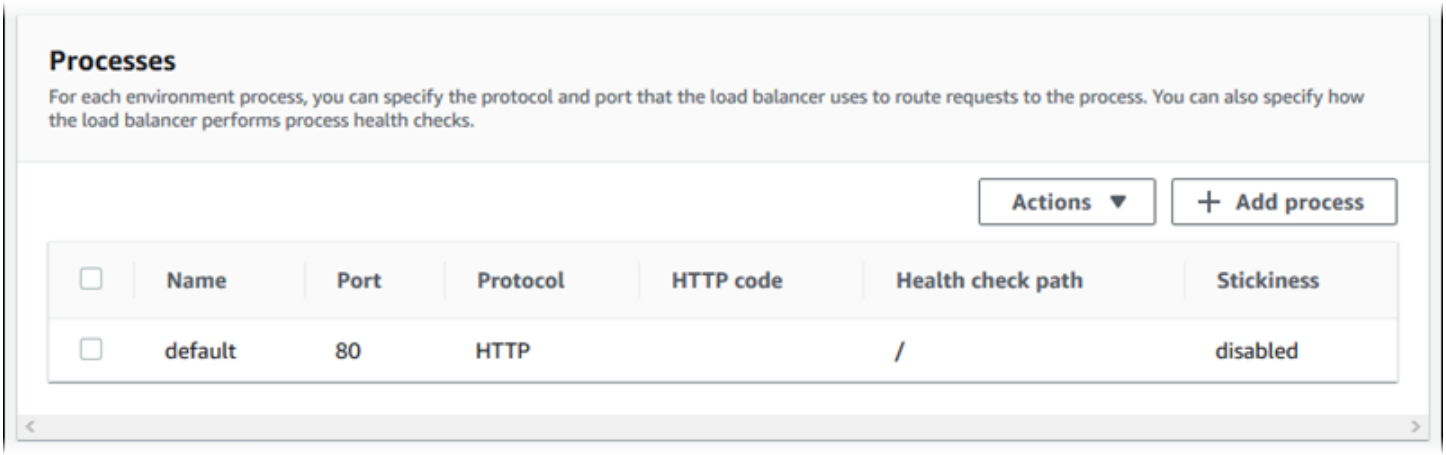
- Load balancer ARN (负载均衡器 ARN) – 要在此环境中使用的共享的负载均衡器。从负载均衡器列表中进行选择，或输入负载均衡器 Amazon Resource Name (ARN)。

- Default listener port (默认侦听器端口) – 共享的负载均衡器所侦听的侦听器端口。从现有侦听器端口列表中进行选择。来自此侦听器的流量（在主机标头中具有环境的 CNAME）将路由到此环境中的默认进程。



进程

使用该列表为共享的负载均衡器指定进程。进程是侦听器将流量路由到的目标。最初，列表显示默认进程，该进程从默认侦听器接收流量。



配置现有进程

1. 选中表条目旁边的复选框，然后选择操作和编辑。
2. 使用环境进程对话框编辑设置，然后选择保存。

添加进程

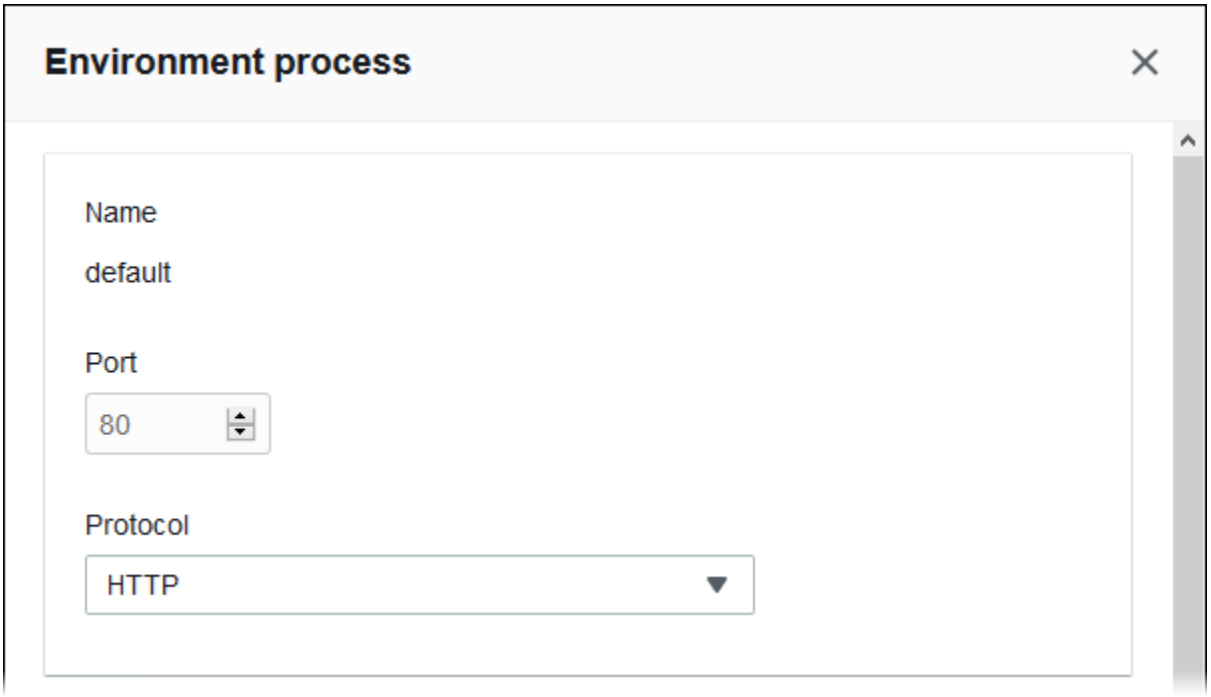
1. 选择添加进程。
2. 在环境进程对话框中，配置所需的设置，然后选择添加。

Application Load Balancer 的环境进程对话框设置

- 定义
- 运行状况检查
- 会话

定义

可以使用以下设置定义进程：其名称以及它在其中侦听请求的端口和协议。

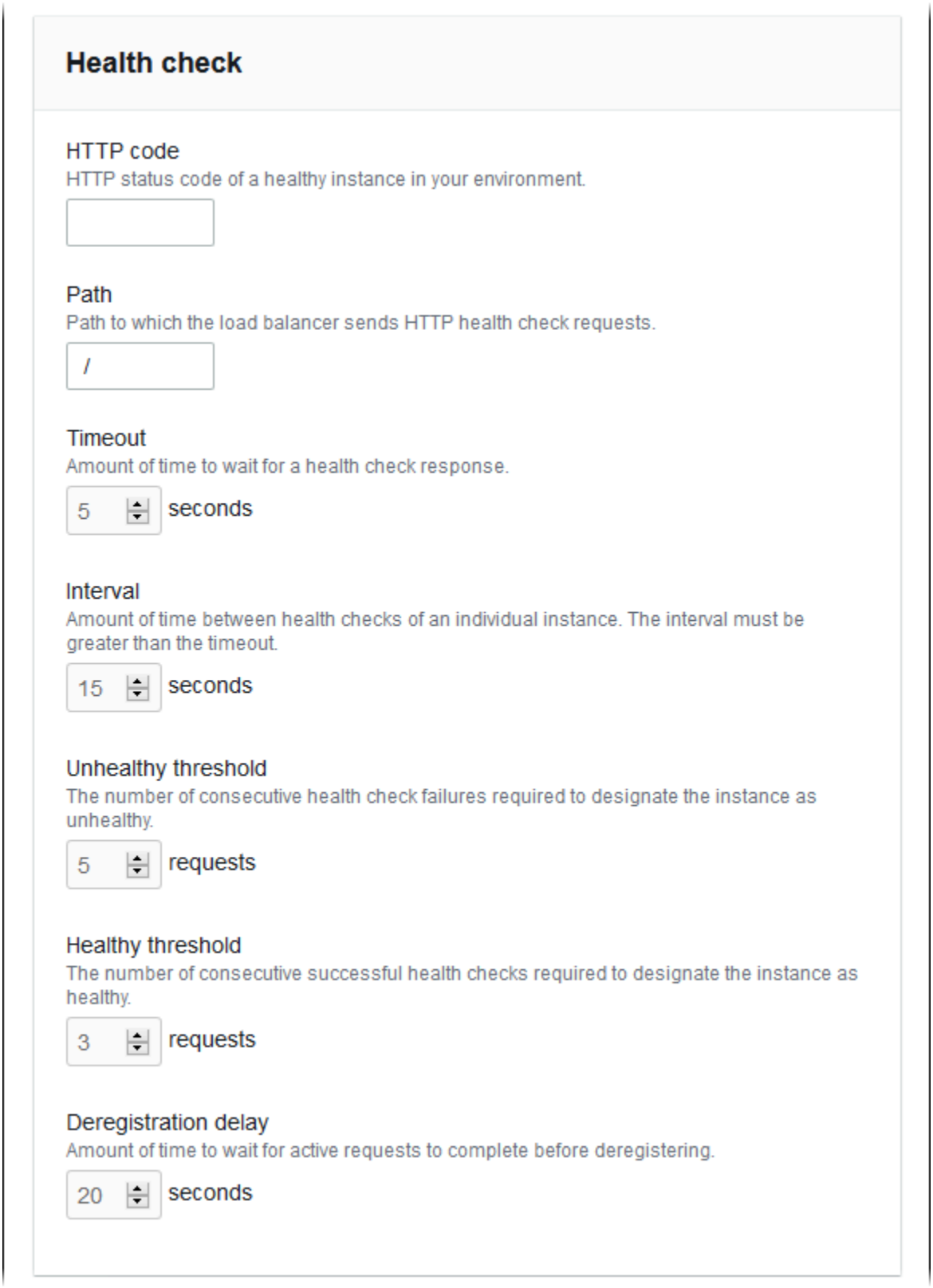


运行状况检查

可以使用以下设置配置进程运行状况检查：

- HTTP code (HTTP 代码) – 指定正常运行的进程的 HTTP 状态代码。
- Path (路径) – 进程的运行状况检查请求路径。
- Timeout (超时) – 等待运行状况检查响应的时间（秒）。

- Interval (间隔) – 单个实例的两次运行状况检查间隔的时间（秒）。间隔必须大于超时。
- Unhealthy threshold (不正常阈值) 和 Healthy threshold (正常阈值) – 在 Elastic Load Balancing 更改实例的运行状况之前，实例必须通过或未通过的运行状况检查次数。
- Deregistration delay (取消注册延迟) – 在取消注册实例之前等待活动请求完成的时间（秒）。

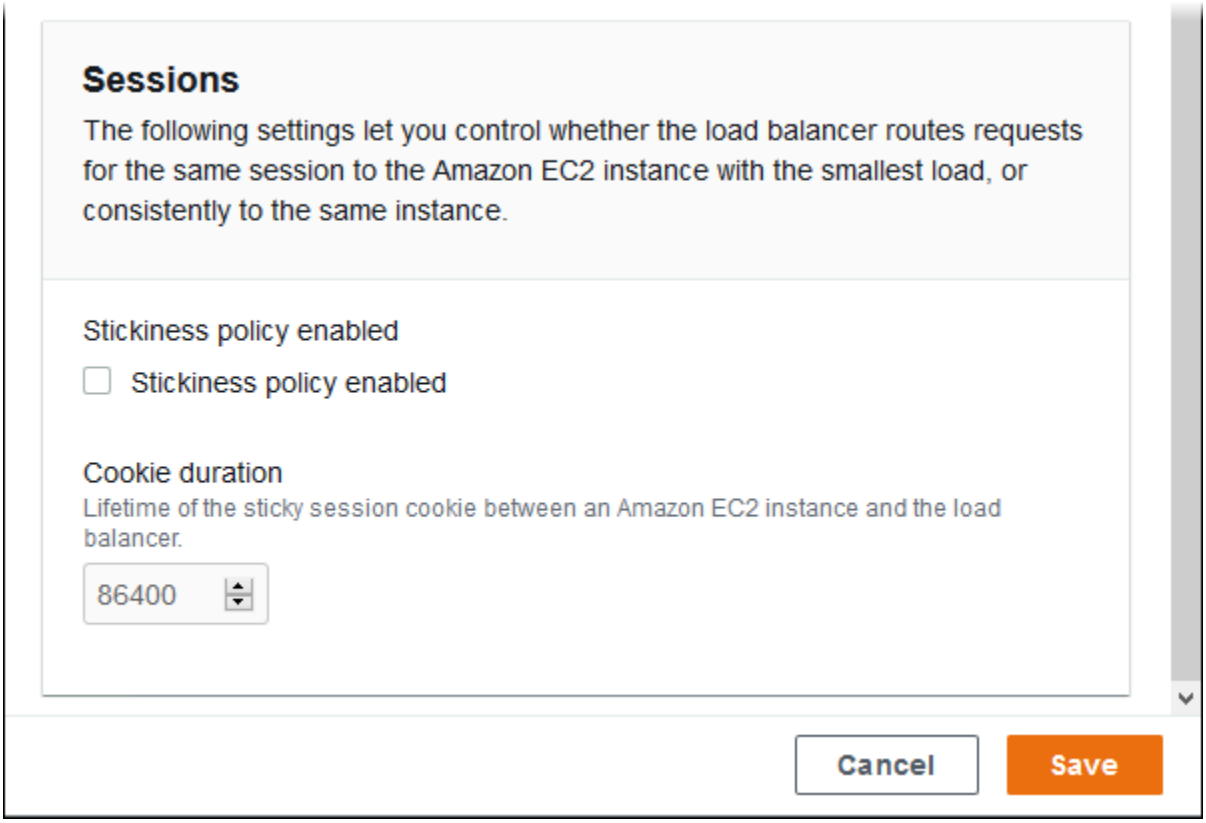

Health check
HTTP code
HTTP status code of a healthy instance in your environment.
Path
Path to which the load balancer sends HTTP health check requests.
/
Timeout
Amount of time to wait for a health check response.
5 seconds
Interval
Amount of time between health checks of an individual instance. The interval must be greater than the timeout.
15 seconds
Unhealthy threshold
The number of consecutive health check failures required to designate the instance as unhealthy.
5 requests
Healthy threshold
The number of consecutive successful health checks required to designate the instance as healthy.
3 requests
Deregistration delay
Amount of time to wait for active requests to complete before deregistering.
20 seconds

> **Note**
>
> Elastic Load Balancing 运行状况检查不会影响环境的 Auto Scaling 组的运行状况检查行为。除非您手动配置了 Amazon EC2 Auto Scaling 进行替换，否则 Amazon EC2 Auto Scaling 不会自动替换未通过 Elastic Load Balancing 运行状况检查的实例。有关详细信息，请参阅 Auto Scaling 运行状况检查设置。

有关运行状况检查以及其对环境的总体运行状况的影响的更多信息，请参阅基本运行状况报告。

会话

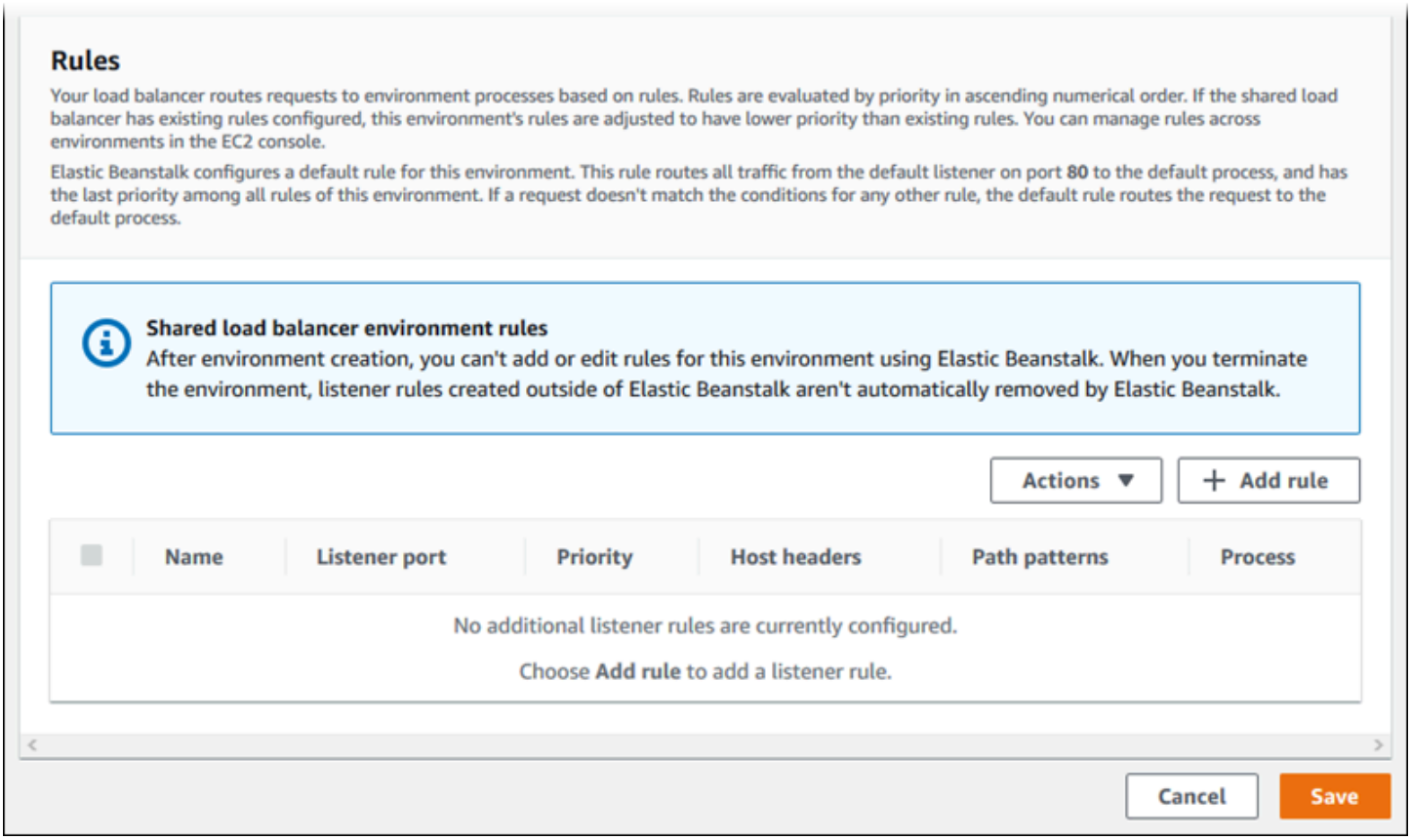
选中或清除粘性策略已启用框来启用或禁用粘性会话。使用 Cookie 持续时间配置粘性会话的持续时间，最多为 **604800** 秒。



规则

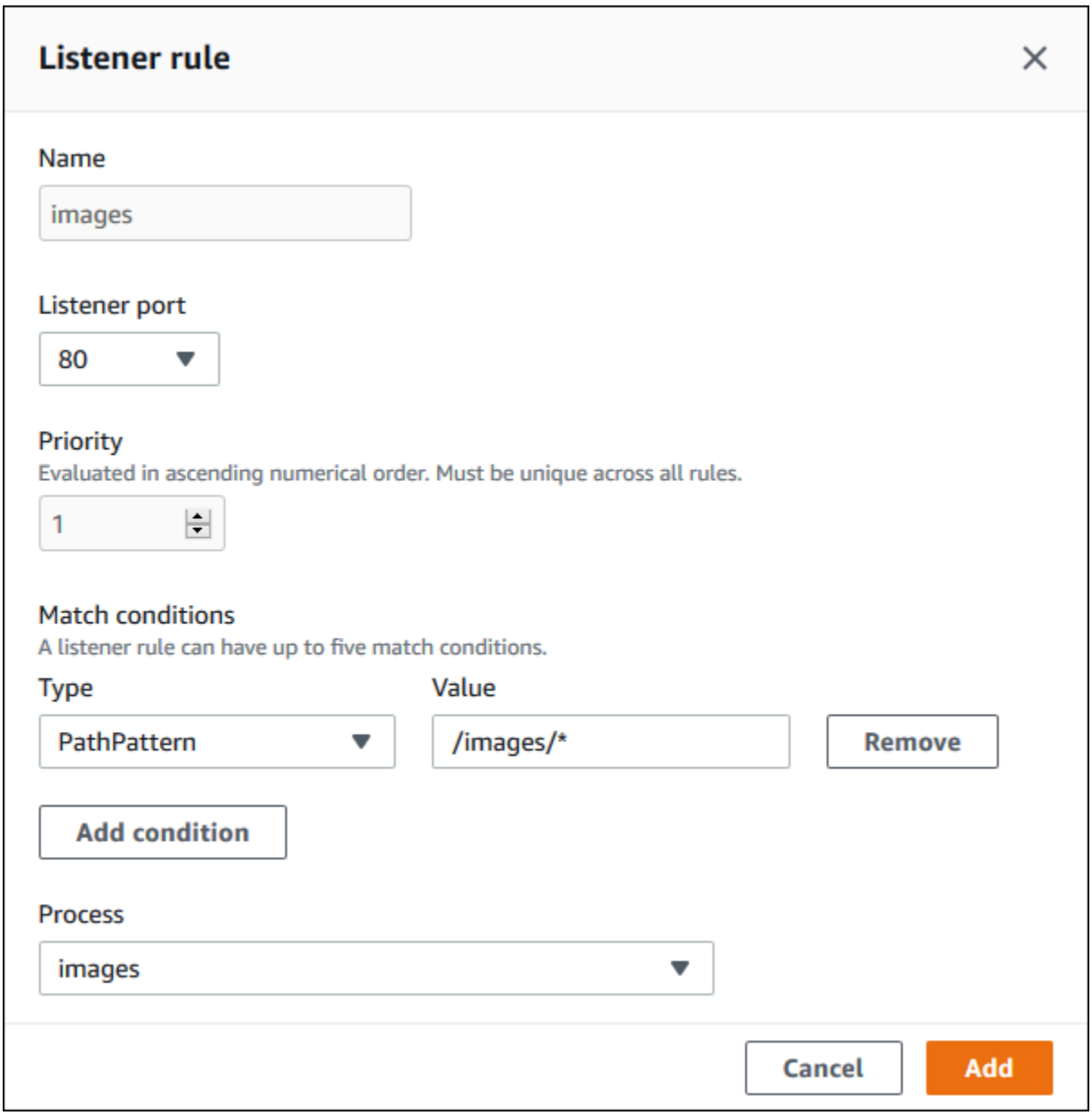
使用该列表为共享的负载均衡器指定自定义侦听器规则。规则将侦听器在特定路径模式上接收的请求映射到目标进程。每个侦听器可以有多个规则，将不同路径上的请求路由到共享侦听器的不同环境的实例上的不同进程。

规则具有数字优先级，它确定将规则应用于传入请求的优先顺序。Elastic Beanstalk 添加了一个默认规则，该规则将所有默认侦听器的流量路由到新环境的默认进程。默认规则的优先顺序最低；如果同一侦听器没有与传入请求匹配的任何其他规则，则应用该规则。起初在您还未添加自定义规则时，该列表是空的。不显示默认规则。



您可以编辑现有规则的设置或添加新的规则。要在列表中开始编辑或添加规则，请使用为进程列表列出的相同步骤。将打开侦听器规则对话框并包含以下设置：

- Name (名称) – 规则的名称。
- Listener port (侦听器端口) – 规则应用到的侦听器端口。
- Priority (优先级) – 规则的优先级。优先级序号越小，优先顺序越靠前。侦听器的规则优先级必须是唯一的。Elastic Beanstalk 将规则优先级视为跨共享环境的相对优先级，并在创建过程中将其映射到绝对优先级。
- Match conditions (匹配条件) – 规则应用到的请求 URL 条件的列表。有两种类型的条件：HostHeader（URL 的域部分）和 PathPattern（URL 的路径部分）。为环境子域保留了一个条件，您最多可以添加四个条件。每个条件值的长度最多为 128 个字符，并且可以包含通配符。
- Process (进程) – 负载均衡器将与规则匹配的请求路由到的进程。



示例：将共享 Application Load Balancer 用于基于微服务的安全应用程序

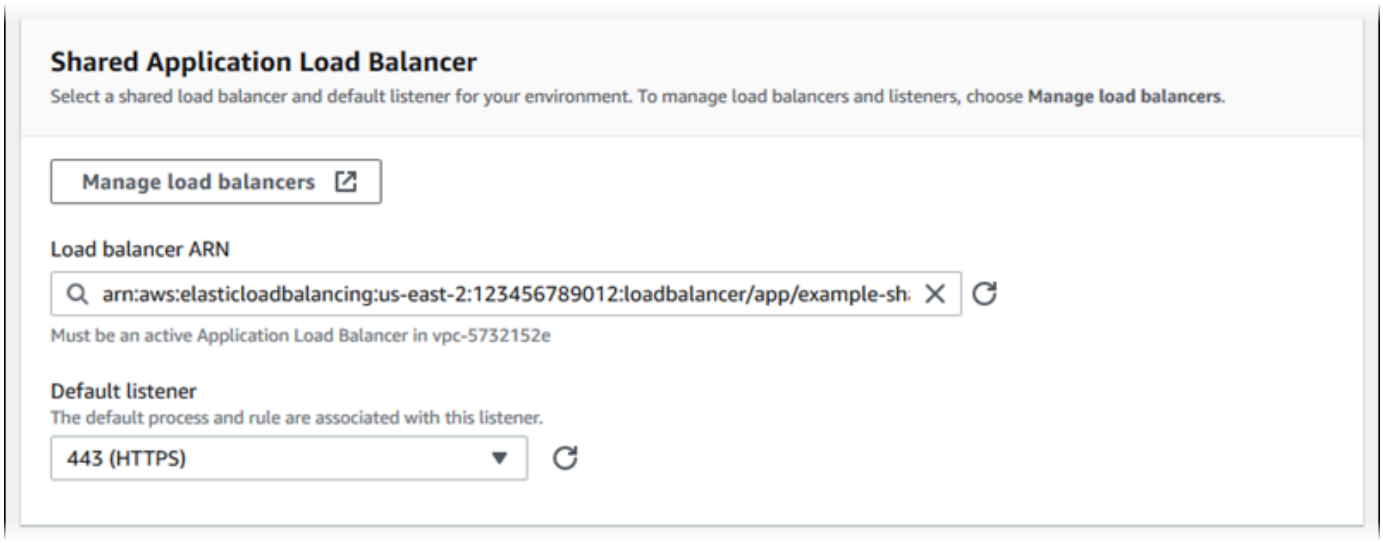
在此示例中，您的应用程序由几个微服务组成，每个微服务都作为一个 Elastic Beanstalk 环境实现。此外，您需要端到端流量加密。我们将演示其中一个微服务环境，该环境具有用于用户请求的主流程和用于处理管理请求的单独流程。

要满足这些要求，请使用 Amazon EC2 创建将在微服务之间共享的 Application Load Balancer。在端口 443 和 HTTPS 协议上添加安全侦听器。然后，将多个 SSL 证书添加到侦听器，每个微服务域一个。有关创建应用程序负载均衡器和安全侦听器的详细信息，请参阅《应用程序负载均衡器用户指南》中的创建应用程序负载均衡器和为应用程序负载均衡器创建 HTTPS 侦听器。

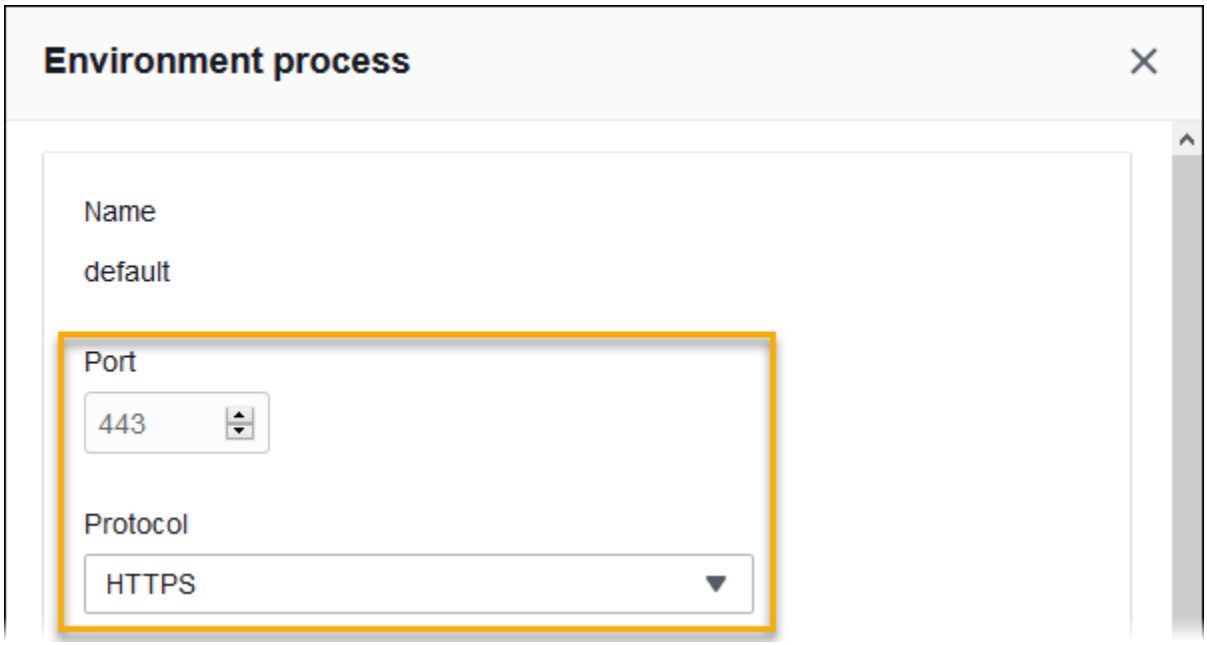
在 Elastic Beanstalk 中，将每个微服务环境配置为使用共享 Application Load Balancer，并将默认侦听器端口设置为 443。对于我们在此演示的特定环境，指示默认进程侦听端口 443 上的 HTTPS，并为不同路径上的管理员流量添加进程和侦听器规则。

为该示例配置共享的负载均衡器

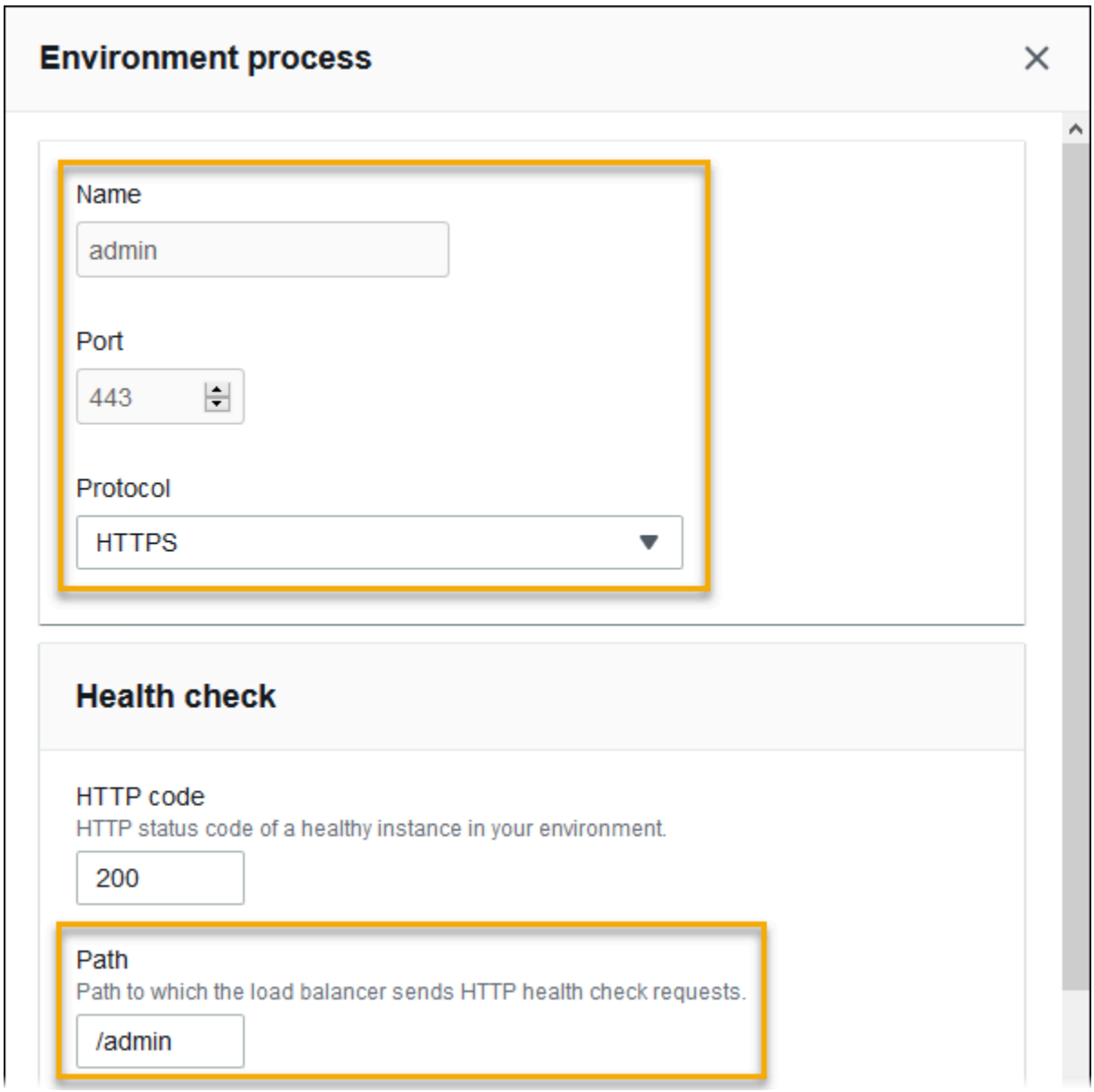
1. 在 Shared Application Load Balancer（共享 Application Load Balancer）部分中，选择您的负载均衡器，对于 Default listener port（默认侦听器端口），选择 **443**。如果侦听器端口是负载均衡器拥有的唯一侦听器，则应该已选择该端口。



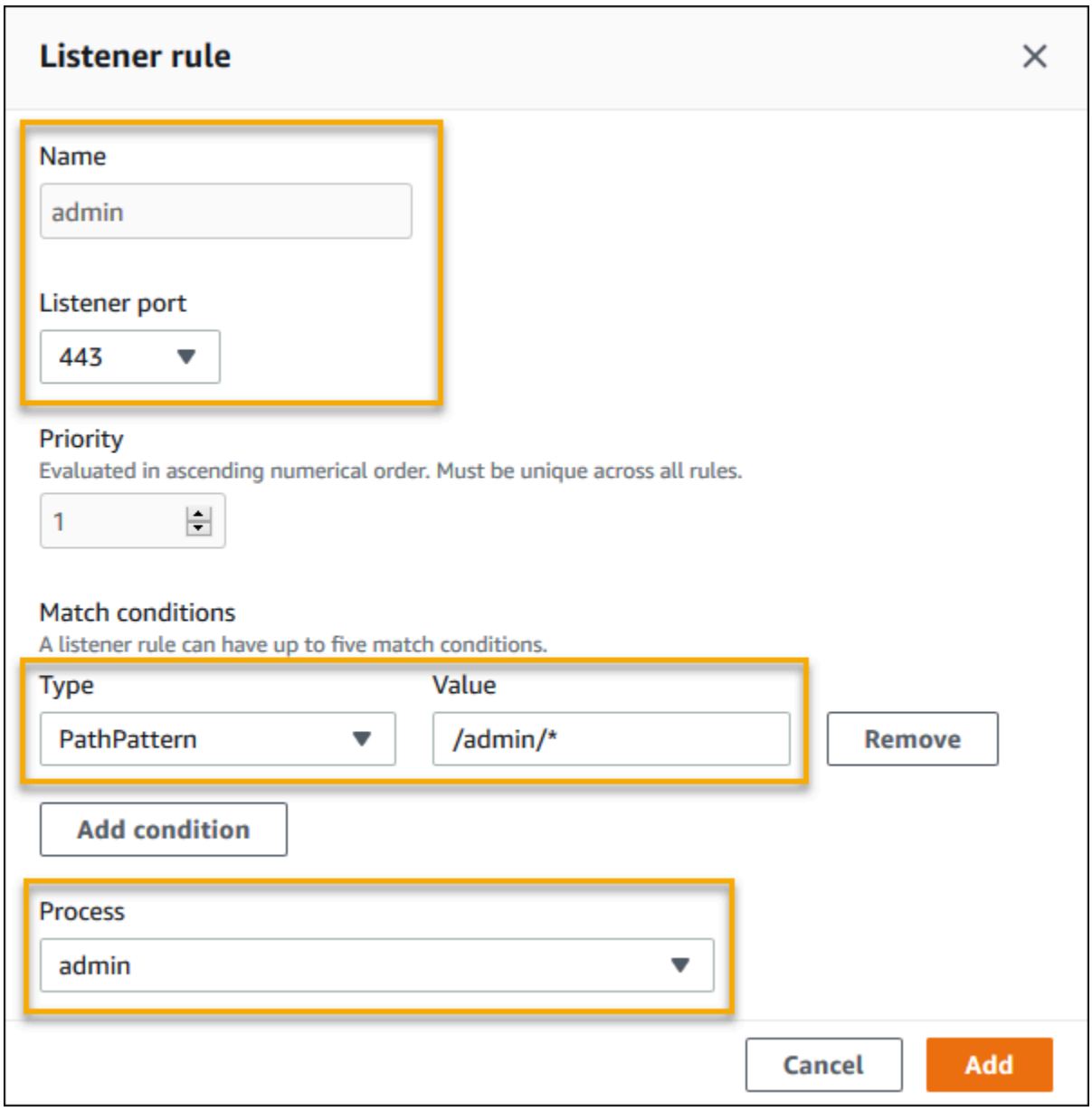
2. 将默认进程配置为 HTTPS。选择默认进程，然后为操作选择编辑。对于端口，输入 **443**。对于协议，选择 **HTTPS**。



3. 添加一个管理进程。对于 Name（名称），请输入 **admin**。对于端口，输入 **443**。对于协议，选择 **HTTPS**。在运行状况检查下，为路径键入 **/admin**。



4. 为管理流量添加一个规则。对于 Name（名称），请输入 **admin**。对于侦听器端口，输入 **443**。对于匹配条件，添加带有 **/admin/*** 值的 PathPattern。对于进程，选择 **admin**。



使用 EB CLI 配置共享 Application Load Balancer

当您运行 eb create 时，EB CLI 会提示您选择负载均衡器类型。如果您选择 `application`（默认设置），并且您的账户至少有一个可共享的 Application Load Balancer，EB CLI 还会询问您是否要使用共享的 Application Load Balancer。如果回答 **y**，还会提示您选择负载均衡器和默认端口。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF
```

```
Select a load balancer type
1) classic
2) application
3) network
(default is 2):

Your account has one or more sharable load balancers. Would you like your new
 environment to use a shared load balancer?(y/N) y

Select a shared load balancer
1)MySharedALB1 - arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/
MySharedALB1/6d69caa75b15d46e
2)MySharedALB2 - arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/
MySharedALB2/e574ea4c37ad2ec8
(default is 1): 2

Select a listener port for your shared load balancer
1) 80
2) 100
3) 443
(default is 1): 3
```

还可以使用命令选项指定共享的负载均衡器。

```
$ eb create test-env --elb-type application --shared-lb MySharedALB2 --shared-lb-
port 443
```

共享 Application Load Balancer 命名空间

您可以在以下命名空间中找到与共享 Application Load Balancer 相关的设置：

- aws:elasticbeanstalk:environment – 选择环境的负载均衡器类型，并告知 Elastic Beanstalk 您将使用共享的负载均衡器。

  无法在配置文件 (.Ebextensions) 中设置这两个选项。
- aws:elbv2:loadbalancer – 配置共享 Application Load Balancer ARN 和安全组。
- aws:elbv2:listener – 通过列出侦听器规则将共享 Application Load Balancer 的侦听器与环境进程相关联。
- aws:elbv2:listenerrule – 配置根据请求路径将流量路由到不同进程的侦听器规则。规则对于 Application Load Balancer 是唯一的，无论是专用还是共享。

- aws:elasticbeanstalk:environment:process – 配置运行状况检查并为在您的环境的实例上运行的过程指定端口和协议。

Example .ebextensions/application-load-balancer-shared.config

要开始使用共享 Application Load Balancer，请使用 Elastic Beanstalk 控制台、EB CLI 或 API 将负载均衡器类型设置为 application，然后选择使用共享的负载均衡器。使用配置文件以配置共享的负载均衡器。

```
option_settings:
  aws:elbv2:loadbalancer:
    SharedLoadBalancer: arn:aws:elasticloadbalancing:us-
east-2:123456789012:loadbalancer/app/MySharedALB2/e574ea4c37ad2ec8
```

> **Note**
>
> 您只能在创建环境时配置此选项。

Example .ebextensions/alb-shared-secure-listener.config

以下配置文件为共享的负载均衡器选择端口 443 上的默认安全侦听器，并将默认进程设置为侦听端口 443。

```
option_settings:
  aws:elbv2:loadbalancer:
    SharedLoadBalancer: arn:aws:elasticloadbalancing:us-
east-2:123456789012:loadbalancer/app/MySharedALB2/e574ea4c37ad2ec8
  aws:elbv2:listener:443:
    rules: default
  aws:elasticbeanstalk:environment:process:default:
    Port: '443'
    Protocol: HTTPS
```

Example .ebextensions/alb-shared-admin-rule.config

以下配置文件在上一个示例的基础上构建，并添加了一条规则，该规则将请求路径为 /admin 的流量路由到一个名为 admin 的进程，该进程在端口 4443 上进行侦听。

```
option_settings:
```

```
  aws:elbv2:loadbalancer:
    SharedLoadBalancer: arn:aws:elasticloadbalancing:us-
east-2:123456789012:loadbalancer/app/MySharedALB2/e574ea4c37ad2ec8
  aws:elbv2:listener:443:
    rules: default,admin
  aws:elasticbeanstalk:environment:process:default:
    Port: '443'
    Protocol: HTTPS
  aws:elasticbeanstalk:environment:process:admin:
    HealthCheckPath: /admin
    Port: '4443'
    Protocol: HTTPS
  aws:elbv2:listenerrule:admin:
    PathPatterns: /admin/*
    Priority: 1
    Process: admin
```

配置 Network Load Balancer

在启用负载均衡时，将为您的 AWS Elastic Beanstalk 环境配备 Elastic Load Balancing 负载均衡器，以便在您的环境中的实例之间分配流量。Elastic Load Balancing 支持多种负载均衡器类型。要了解更多信息，请参阅 Elastic Load Balancing 用户指南。Elastic Beanstalk 可以为您创建负载均衡器，或者让您指定已创建的共享负载均衡器。

本主题介绍 Elastic Beanstalk 创建并专用于您的环境的 Network Load Balancer 的配置。有关配置 Elastic Beanstalk 支持的所有负载均衡器类型的信息，请参阅Elastic Beanstalk 环境的负载均衡器。

> **Note**
>
> 您只能在创建环境期间选择您的环境使用的负载均衡器类型。您可以更改设置以管理运行的环境的负载均衡器行为，但不能更改其类型。

介绍

通过 Network Load Balancer，默认侦听器接受端口 80 上的 TCP 请求并将它们分配到环境中的实例。您可以配置运行状况检查行为、配置侦听器端口或在其他端口中添加侦听器。

> **Note**
>
> 与经典负载均衡器或 Application Load Balancer 不同，Network Load Balancer 不能有应用层（第 7 层）HTTP 或 HTTPS 侦听器。它仅支持传输层（第 4 层）TCP 侦听器。HTTP 和 HTTPS 流量可通过 TCP 路由到您的环境。要在 Web 客户端与您的环境之间建立安全的 HTTPS 连接，请在该环境的实例上安装自签名证书，并将该实例配置为在适当端口（通常为 443）上侦听并终止 HTTPS 连接。此配置根据不同平台而有所变化。有关说明，请参阅配置应用程序以终止实例的 HTTPS 连接；然后，配置您的 Network Load Balancer 以添加一个侦听器，该侦听器映射到在相应端口上侦听的进程。

Network Load Balancer 支持主动运行状况检查。这些检查基于到根 (/) 路径的消息。此外，Network Load Balancer 支持被动运行状况检查。它会自动检测有故障的后端实例并只将流量路由到正常运行的实例。

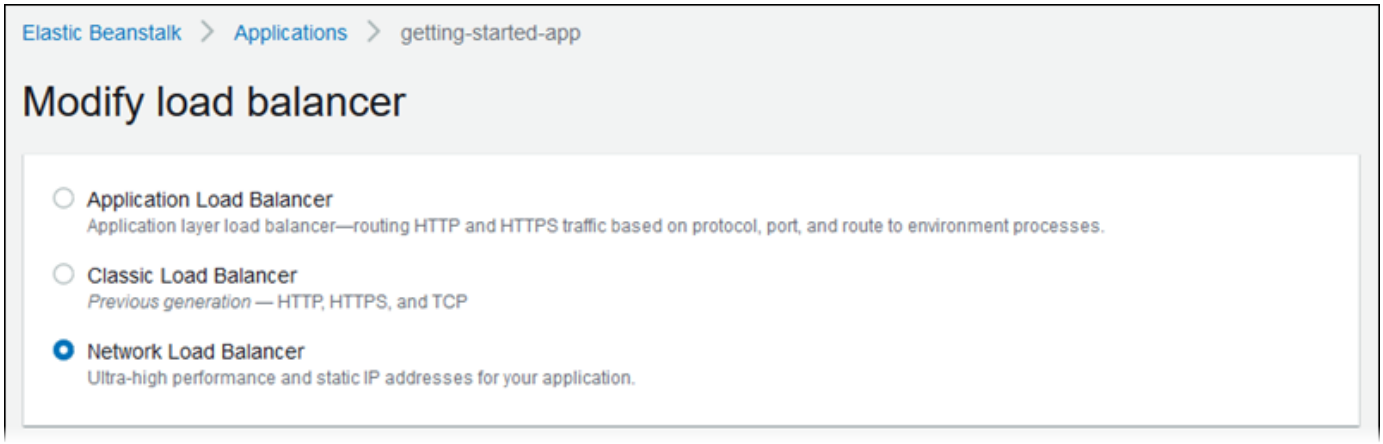
使用 Elastic Beanstalk 控制台配置 Network Load Balancer

在创建环境期间或以后运行您的环境时，您可以使用 Elastic Beanstalk 控制台配置 Network Load Balancer 的侦听器和进程。

环境创建期间，在 Elastic Beanstalk 控制台中配置 Network Load Balancer

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments (环境)。
3. 选择 Create a new environment (创建新环境) 以开始创建环境。
4. 在向导的主页上，在选择创建环境之前，选择配置更多选项。
5. 选择高可用性配置预设。

   或者，在容量配置类别中配置负载均衡环境类型。有关详细信息，请参阅 容量。
6. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。
7. 如果尚未选择 Network Load Balancer 选项，请选择该选项。



8. 根据环境需要进行任何 Network Load Balancer 配置更改。
9. 选择保存，然后进行您的环境所需的任何其他配置更改。
10. 选择 Create environment（创建环境）。

在 Elastic Beanstalk 控制台中配置正在运行环境的 Network Load Balancer

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。

> **Note**
>
> 如果 Load balancer (负载均衡器) 配置类别没有 Edit (编辑) 按钮，则表示您的环境没有负载均衡器。要了解如何设置负载均衡器，请参阅更改环境类型。

5. 根据环境需要进行 Network Load Balancer 配置更改。
6. 要保存更改，请选择页面底部的 Apply（应用）。

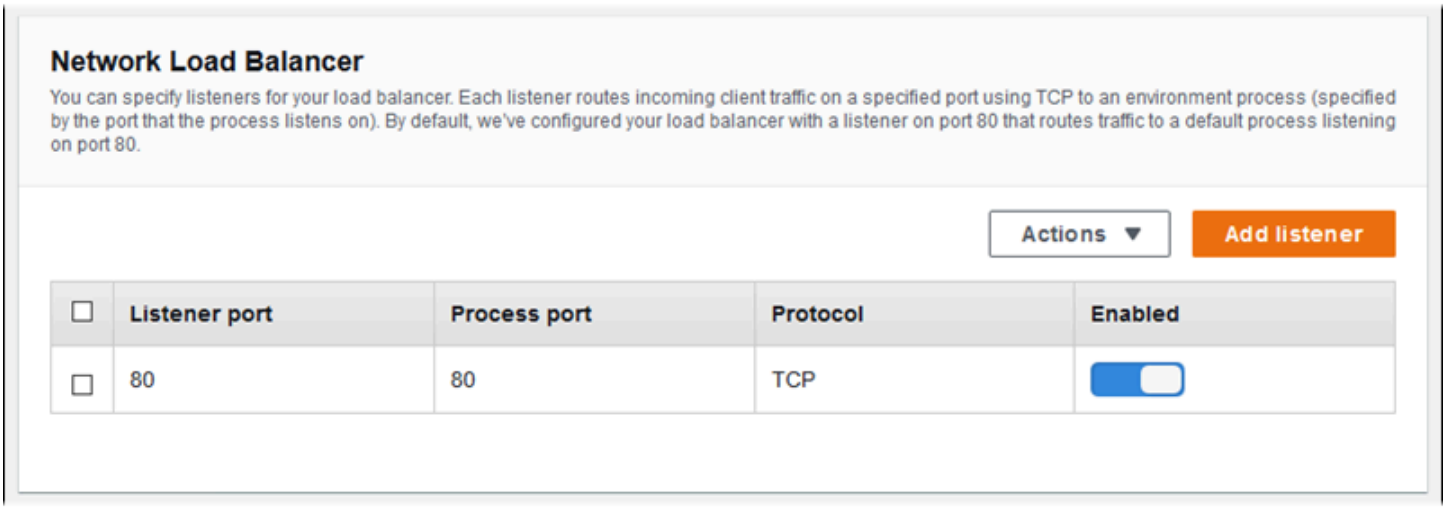
Network Load Balancer 支持

- 侦听器

- 进程

侦听器

可以使用该列表为您的负载均衡器指定侦听器。每个侦听器将指定端口上的传入客户端流量路由到您的实例上的进程。最初，该列表显示默认侦听器，它将端口 80 上的传入 HTTP 流量路由到名为 default 的进程，该进程侦听端口 80。



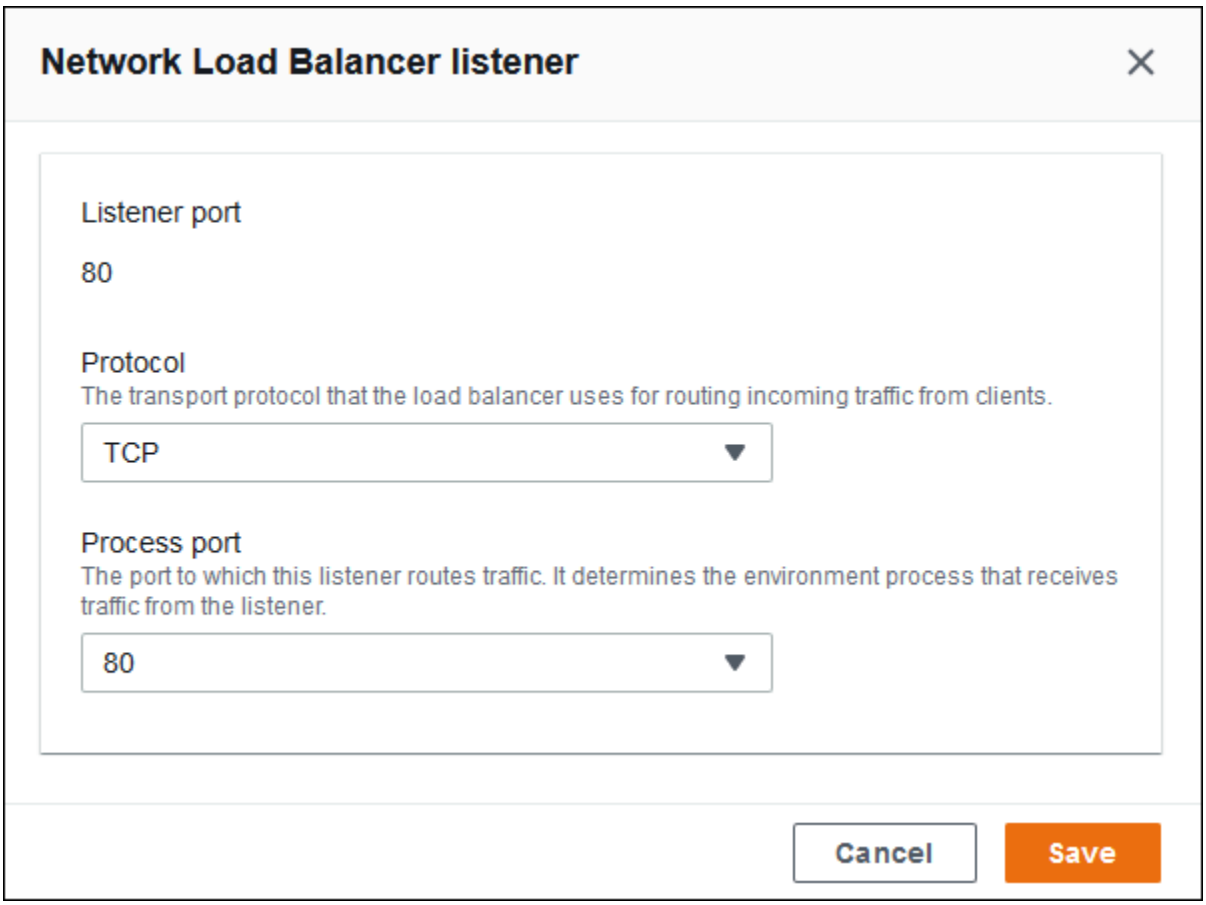
配置现有的侦听器

1. 选中表条目旁边的复选框，然后选择操作和编辑。
2. 使用 Network Load Balancer listener (网络负载均衡器侦听器) 对话框编辑设置，然后选择 Save (保存)。

添加侦听器

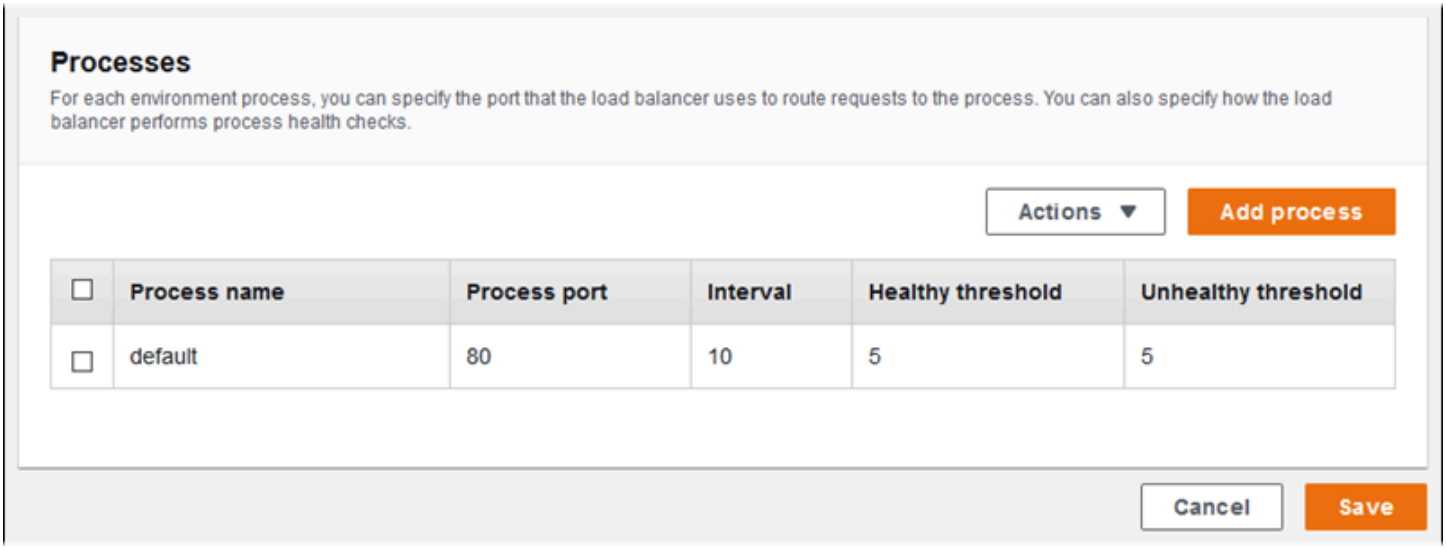
1. 选择 Add listener (添加侦听器)。
2. 在 Network Load Balancer listener (网络负载均衡器侦听器) 对话框中，配置所需的设置，然后选择 Add (添加)。

使用 Network Load Balancer listener (网络负载均衡器侦听器) 对话框来配置侦听器侦听流量所在的端口，并选择您要将流量路由到的进程（通过进程侦听的端口指定）。



进程

可以使用该列表为您的负载均衡器指定进程。进程是侦听器将流量路由到的目标。每个侦听器将指定端口上的传入客户端流量路由到您的实例上的进程。最初，该列表显示默认进程，它侦听端口 80 上的传入流量。



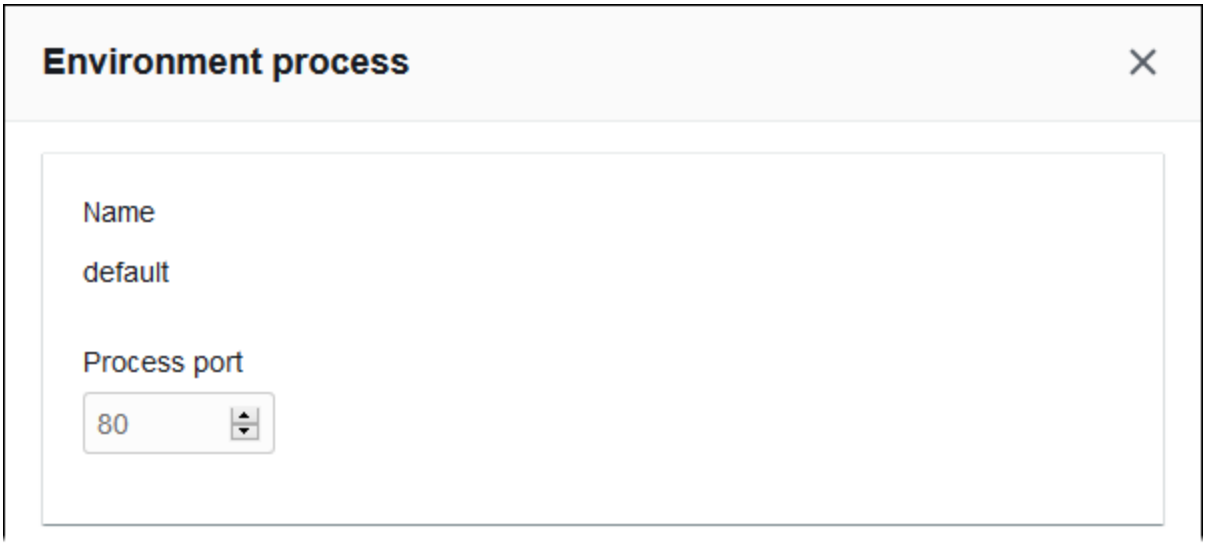
您可以编辑现有进程的设置或添加新的进程。要在列表中开始编辑或添加进程，请使用为侦听器列表列出的相同步骤。将打开环境进程对话框。

Network Load Balancer 的环境进程对话框设置

- 定义
- 运行状况检查

定义

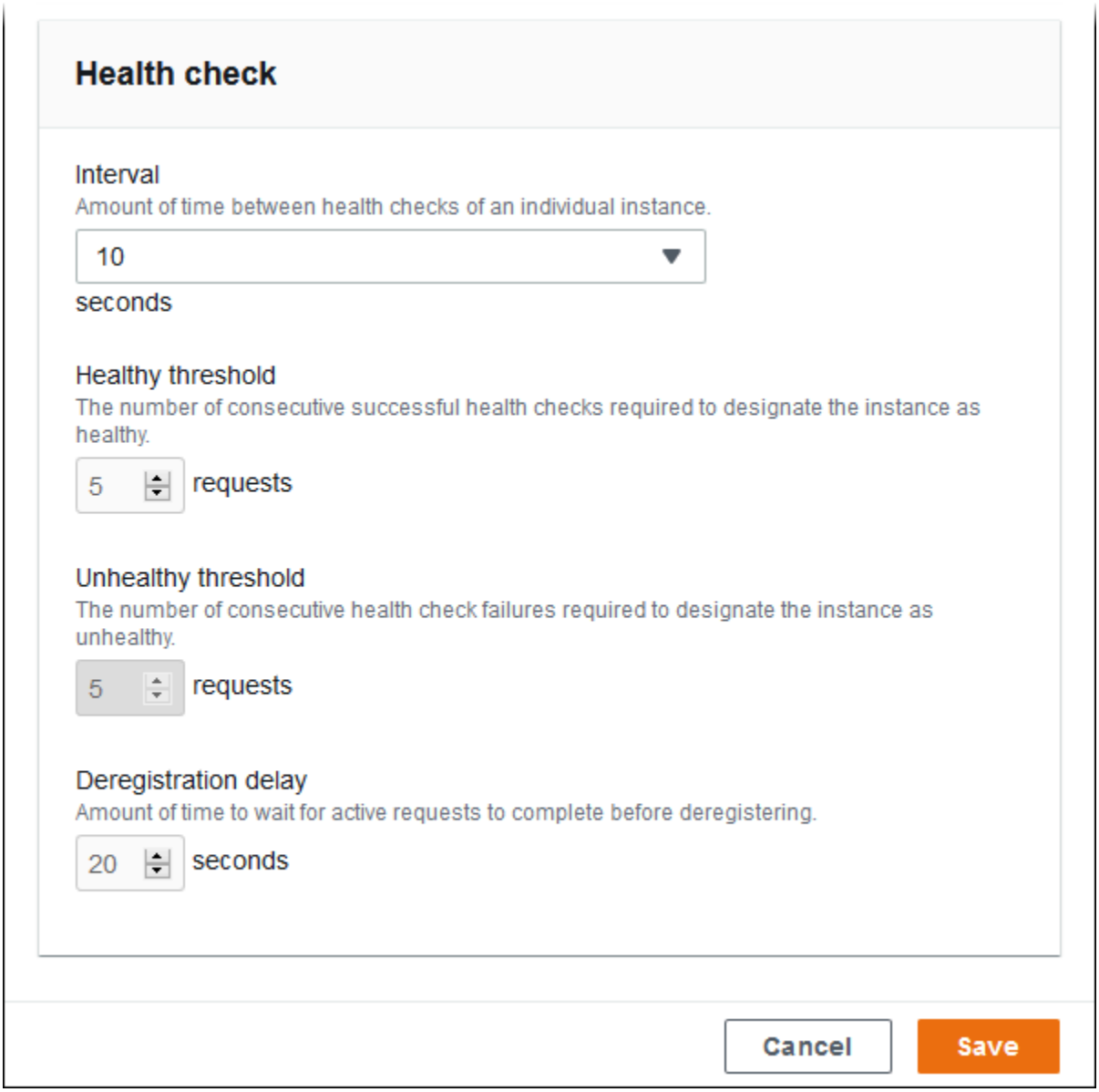
可以使用以下设置定义进程：其 Name (名称) 以及它在其中侦听请求的 Process port (进程端口)。



运行状况检查

可以使用以下设置配置进程运行状况检查：

- Interval (间隔) – 单个实例的两次运行状况检查间隔的时间（秒）。
- Healthy threshold (正常阈值) – 在 Elastic Load Balancing 更改实例的运行状况之前，实例必须通过或未通过的运行状况检查次数。（对于 Network Load Balancer，Unhealthy threshold (不正常阈值) 是始终等于正常阈值的只读设置。）
- Deregistration delay (取消注册延迟) – 在取消注册实例之前等待活动请求完成的时间（秒）。



> **Note**
>
> Elastic Load Balancing 运行状况检查不会影响环境的 Auto Scaling 组的运行状况检查行为。除非您手动配置了 Amazon EC2 Auto Scaling 进行替换，否则 Amazon EC2 Auto Scaling 不会自动替换未通过 Elastic Load Balancing 运行状况检查的实例。有关详细信息，请参阅Auto Scaling 运行状况检查设置。

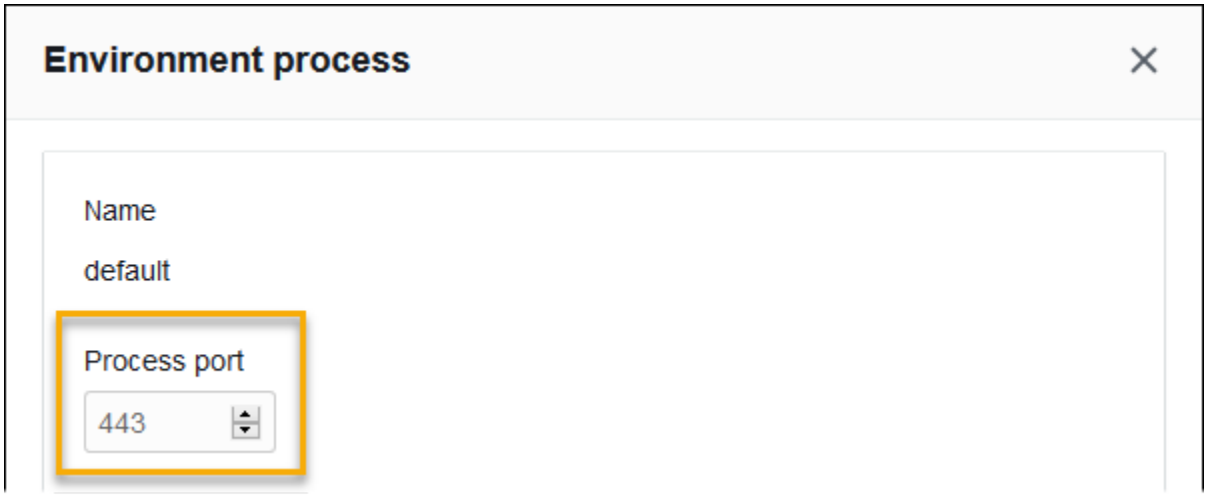
有关运行状况检查以及其对环境的总体运行状况的影响的更多信息，请参阅基本运行状况报告。

示例：用于端对端加密环境的 Network Load Balancer

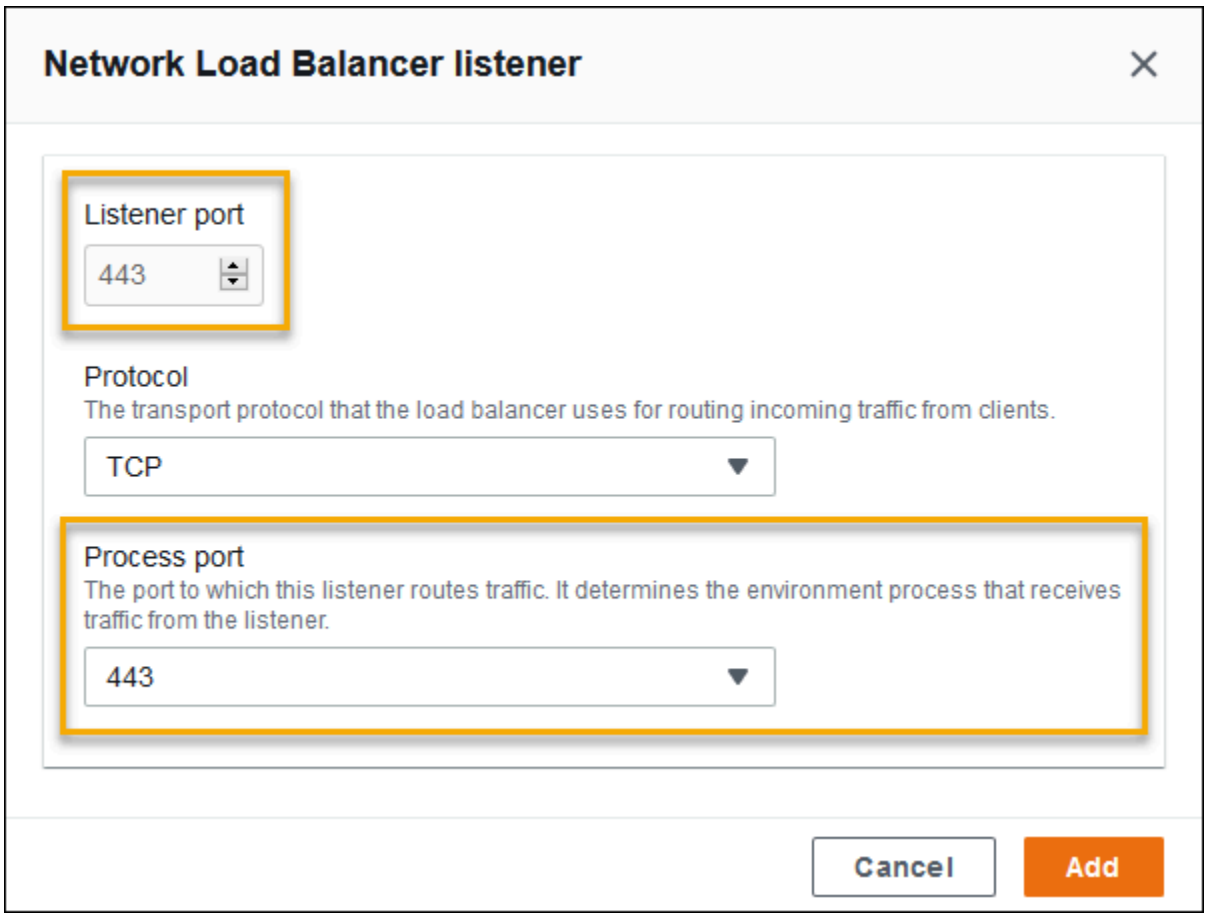
在本示例中，您的应用程序需要端对端流量加密。要配置您环境的 Network Load Balancer 以满足这些要求，您需要将默认进程配置为侦听端口 443、向端口 443 添加一个可将流量路由到默认进程的侦听器，并禁用默认侦听器。

为该示例配置负载均衡器

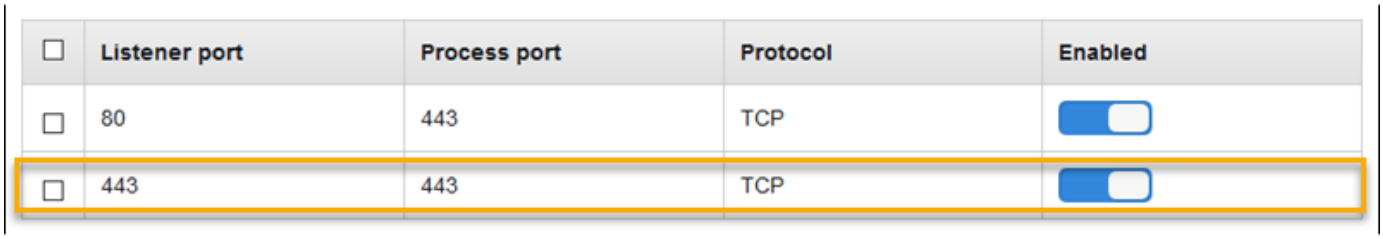
1. 配置默认进程。选择默认进程，然后为操作选择编辑。对于 Process port (进程端口) 键入 443。



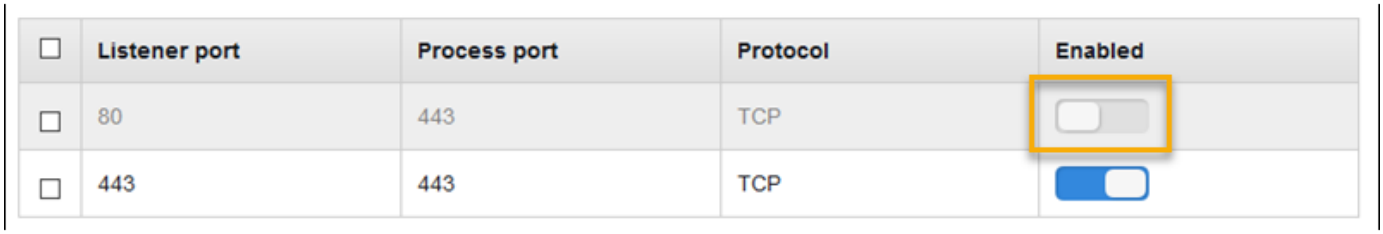
2. 添加端口 443 侦听器。添加新侦听器。对于 Listener port (侦听器端口)，键入 443。对于 Process port (进程端口)，确保 443 已选中。



您现在可以查看列表上的其他侦听器。



3. 禁用默认端口 80 侦听器。对于默认侦听器，关闭已启用选项。



使用 EB CLI 配置 Network Load Balancer

当您运行 eb create 时，EB CLI 会提示您选择负载均衡器类型。

```
$ eb create
Enter Environment Name
(default is my-app): test-env
Enter DNS CNAME prefix
(default is my-app): test-env-DLW24ED23SF

Select a load balancer type
1) classic
2) application
3) network
(default is 1): 3
```

您也可以使用 `--elb-type` 选项指定负载均衡器类型。

```
$ eb create test-env --elb-type network
```

Network Load Balancer 命名空间

您可以在以下命名空间中找到与 Network Load Balancer 相关的设置：

- `aws:elasticbeanstalk:environment` – 选择用于环境的负载均衡器类型。Network Load Balancer 的值为 `network`。
- `aws:elbv2:listener` – 在 Network Load Balancer 上配置侦听器。这些设置将映射到经典负载均衡器的 `aws:elb:listener` 中的设置。
- `aws:elasticbeanstalk:environment:process` – 配置运行状况检查并为在您的环境的实例上运行的过程指定端口和协议。端口和协议设置将映射到经典负载均衡器上侦听器的 `aws:elb:listener` 中的实例端口和实例协议设置。运行状况检查设置将映射到 `aws:elb:healthcheck` 和 `aws:elasticbeanstalk:application` 命名空间中的设置。

Example .ebextensions/network-load-balancer.config

要开始使用 Network Load Balancer，请使用配置文件将负载均衡器类型设置为 `network`。

```
option_settings:
  aws:elasticbeanstalk:environment:
    LoadBalancerType: network
```

> **Note**
>
> 您只能在环境创建期间设置负载均衡器类型。

Example .ebextensions/nlb-default-process.config

以下配置文件将修改默认进程的运行状况检查设置。

```
option_settings:
  aws:elasticbeanstalk:environment:process:default:
    DeregistrationDelay: '20'
    HealthCheckInterval: '10'
    HealthyThresholdCount: '5'
    UnhealthyThresholdCount: '5'
    Port: '80'
    Protocol: TCP
```

Example .ebextensions/nlb-secure-listener.config

以下配置文件将为端口 443 上的安全流量添加一个侦听器和用于侦听端口 443 的匹配的目标过程。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
```

`DefaultProcess` 选项以此方式命名是因为 Application Load Balancer，它可能在同一端口上具有非默认侦听器，以用于侦听到特定路径的流量（如需详细信息，请参阅 应用程序负载均衡器）。对于 Network Load Balancer，该选项为该侦听器指定唯一的目标进程。

在此示例中，我们将进程命名为 `https`，因为它侦听安全 (HTTPS) 流量。由于 Network Load Balancer 只能使用 TCP，因此该侦听器使用 TCP 协议将流量发送到指定端口上的进程。由于 HTTP 和 HTTPS 网络流量是在 TCP 上实施的，因此，可以这样做。

配置访问日志

您可以使用配置文件将环境的负载均衡器配置为将访问日志上传到 Amazon S3 存储桶。有关说明，请参阅 GitHub 中的以下示例配置文件：

- loadbalancer-accesslogs-existingbucket.config – 将负载均衡器配置为将访问日志上传到现有 Amazon S3 存储桶。
- loadbalancer-accesslogs-newbucket.config – 将负载均衡器配置为将访问日志上传到新的存储桶。

将数据库添加到 Elastic Beanstalk 环境

Elastic Beanstalk 提供了与 Amazon Relational Database Service（Amazon RDS）的集成。在创建环境时，您可以使用 Elastic Beanstalk 将 MySQL、PostgreSQL、Oracle 或 SQL Server 数据库添加到现有环境或新环境中。添加数据库实例时，Elastic Beanstalk 会提供应用程序的连接信息。它通过设置数据库主机名、端口、用户名、密码和数据库名称的环境属性来完成此操作。

如果您之前没有将数据库实例与应用程序结合使用，那么我们建议您首先按照本主题中介绍的过程操作，使用 Elastic Beanstalk 服务将数据库添加到测试环境。这样，您就可以验证应用程序是否可以读取环境属性、构建连接字符串以及连接到数据库实例，而无需为 Elastic Beanstalk 外部的数据库执行额外的配置工作。

在验证应用程序可与数据库一起正常使用之后，您可以考虑转向生产环境。此时，您可以选择将数据库与 Elastic Beanstalk 环境解耦，以转向提供更大灵活度的配置。解耦的数据库可以作为外部 Amazon RDS 数据库实例保持运行。解耦数据库不会影响环境的正常运行状态。如果您需要终止环境，您可以这样做，还可以选择在 Elastic Beanstalk 之外保持数据库可用和运行的选项。

使用外部数据库有多种优点。您可以从多个环境连接到外部数据库、使用集成数据库不支持的数据库类型以及执行蓝/绿部署。作为使用 Elastic Beanstalk 创建的解耦数据库的替代方法，您还可以在 Elastic Beanstalk 环境之外创建数据库实例。这两个选项都会产生一个位于 Elastic Beanstalk 环境外部的数据库实例，并且需要其他安全组和连接字符串配置。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

小节目录

数据库生命周期

在将数据库与 Elastic Beanstalk 环境解耦后，您可以选择要对其执行的操作。您可以选择的选项统称为删除策略。在您将数据库与 Elastic Beanstalk 环境解耦或者终止 Elastic Beanstalk 环境之后，以下删除策略适用于数据库：

- Snapshot（快照）– 在 Elastic Beanstalk 终止数据库之前，它会保存数据库的快照。当您将数据库实例添加到 Elastic Beanstalk 环境或创建独立数据库时，您可以从快照中恢复数据库。有关从快照中创建新的独立数据库实例的更多信息，请参阅《Amazon RDS 用户指南》中的从数据库快照还原。您可能产生存储数据库快照的费用。有关更多信息，请参阅 Amazon RDS 定价的备份存储部分。
- Delete（删除）– Elastic Beanstalk 终止数据库。终止后，数据库实例将不再可用于任何操作。
- Retain（保留）– 数据库实例未终止。尽管数据库与 Elastic Beanstalk 解耦，但它仍然可用且可运行。然后，您可以配置一个或多个环境以作为外部 Amazon RDS 数据库实例连接到数据库。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。

使用控制台向环境中添加 Amazon RDS 数据库实例

您可以使用 Elastic Beanstalk 控制台将数据库实例添加到环境。

向环境添加数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 选择数据库引擎，然后输入用户名和密码。
6. 要保存更改，请选择页面底部的 Apply（应用）。

您可以配置以下选项：

- Snapshot（快照）– 选择现有数据库快照。Elastic Beanstalk 还原该快照并将它添加到您的环境。默认值为 None（无）。默认值为 None（无），这将允许您使用此页面上的其他设置来配置新的数据库。
- Engine（引擎）– 选择数据库引擎。
- Engine version（引擎版本）– 选择数据库引擎的具体版本。
- Instance Class（实例类）– 选择数据库实例类。有关数据库实例类的信息，请参阅 https://aws.amazon.com/rds/。
- Storage（存储）– 选择要为数据库预置的存储空间量。您可以以后增加分配的存储空间，但是不能降低它。有关存储分配的信息，请参阅功能。
- Username（用户名）– 仅使用数字和字母组合输入您选择的用户名称。
- Password（密码）– 输入您选择的密码，包含 8-16 个可打印 ASCII 字符（不包括 /、\ 和 @）。
- Availability（可用性）– 选择 High (Multi-AZ)（高(多可用区)）可在第二个可用区中运行热备份以实现高可用性。
- Database deletion policy（数据库删除策略）– 删除策略决定数据库在与您的环境解耦之后会发生的情况。可以设置为下列值之一：`Create Snapshot`、`Retain` 或 `Delete`。这些值在同一主题中的数据库生命周期中进行了描述。

> **Note**
>
> Elastic Beanstalk 会使用您提供的用户名和密码为数据库创建主用户。要了解有关主用户及其权限的更多信息，请参阅主用户账户权限。

添加一个数据库实例大约需要 10 分钟。更新完成后，新数据库与环境耦合。您的应用程序就可以通过以下环境属性访问数据库实例的主机名和其他连接信息。

属性名称	描述	属性值
`RDS_HOSTNAME`	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
`RDS_PORT`	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接

属性名称	描述	属性值
		和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。

连接到数据库

使用连接信息，通过环境变量与应用程序内的数据库连接。有关配合使用 Amazon RDS 和您的应用程序的详细信息，请参阅以下主题。

- Java SE – 连接数据库 (Java SE 平台)
- Java with Tomcat – 连接数据库 (Tomcat 平台)
- Node.js – 连接到数据库
- .NET – 连接到数据库
- PHP – 使用 PDO 或 MySQLi 连接到数据库
- Python – 连接到数据库
- Ruby – 连接到数据库

使用控制台配置集成 RDS 数据库实例

在 Elastic Beanstalk 控制台中，您可以在环境的 Configuration（配置）页面上的 Database（数据库）部分中查看和修改数据库实例的配置设置。

在 Elastic Beanstalk 控制台中配置环境的数据库实例

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。

创建数据库后，您可以修改 Instance class（实例类）、Storage（存储）、Password（密码）、Availability（可用性）和 Database deletion policy（数据库删除策略）设置。如果您更改实例类，Elastic Beanstalk 将重新预置数据库实例。

如果您不再需要 Elastic Beanstalk 将数据库与环境关联，则可以通过选择 Decouple database（解耦数据库）来选择将其解耦。了解此操作涉及的选项和注意事项非常重要。有关更多信息，请参阅the section called “使用控制台解耦 RDS 数据库实例”。

> **警告**
>
> 请勿在 Elastic Beanstalk 提供的功能之外修改耦合数据库实例的设置（例如，在 Amazon RDS 控制台中）。如果您执行此操作，则 Amazon RDS 数据库配置可能与您的环境的定义不同步。在更新或重新启动环境时，环境中指定的设置将覆盖在 Elastic Beanstalk 外部所做的任何设置。
>
> 如果您需要修改 Elastic Beanstalk 不直接支持的设置，请使用 Elastic Beanstalk 配置文件。

使用配置文件配置集成 RDS 数据库实例

您可以使用配置文件配置环境的数据库实例。使用 `aws:rds:dbinstance` 命名空间中的选项。以下示例将分配的数据库存储大小修改为 100 GB。

Example .ebextensions/db-instance-options.config

```
option_settings:
  aws:rds:dbinstance:
```

```
    DBAllocatedStorage: 100
```

如果您需要配置 Elastic Beanstalk 不支持的数据库实例属性，则仍可使用配置文件，并使用 `resources` 键指定您的设置。以下示例将值设置为 `StorageType` 和 `Iops` Amazon RDS 属性。

Example .ebextensions/db-instance-properties.config

```
Resources:
  AWSEBRDSDatabase:
    Type: AWS::RDS::DBInstance
    Properties:
      StorageType:io1
      Iops: 1000
```

使用控制台解耦 RDS 数据库实例

您可以将数据库与 Elastic Beanstalk 环境解耦，而不会影响环境的运行状况。在解耦数据库之前，请考虑以下要求：

- 数据库解耦后应该会发生什么？

  您可以选择创建数据库的快照，然后将其终止，将数据库作为 Elastic Beanstalk 之外的独立数据库保留运行，或者永久删除该数据库。Database deletion policy（数据库删除策略）设置决定了此结果。有关删除策略的详细描述，请参阅在同一主题中的 数据库生命周期。
- 在解耦之前，您是否需要对数据库配置设置进行任何更改？

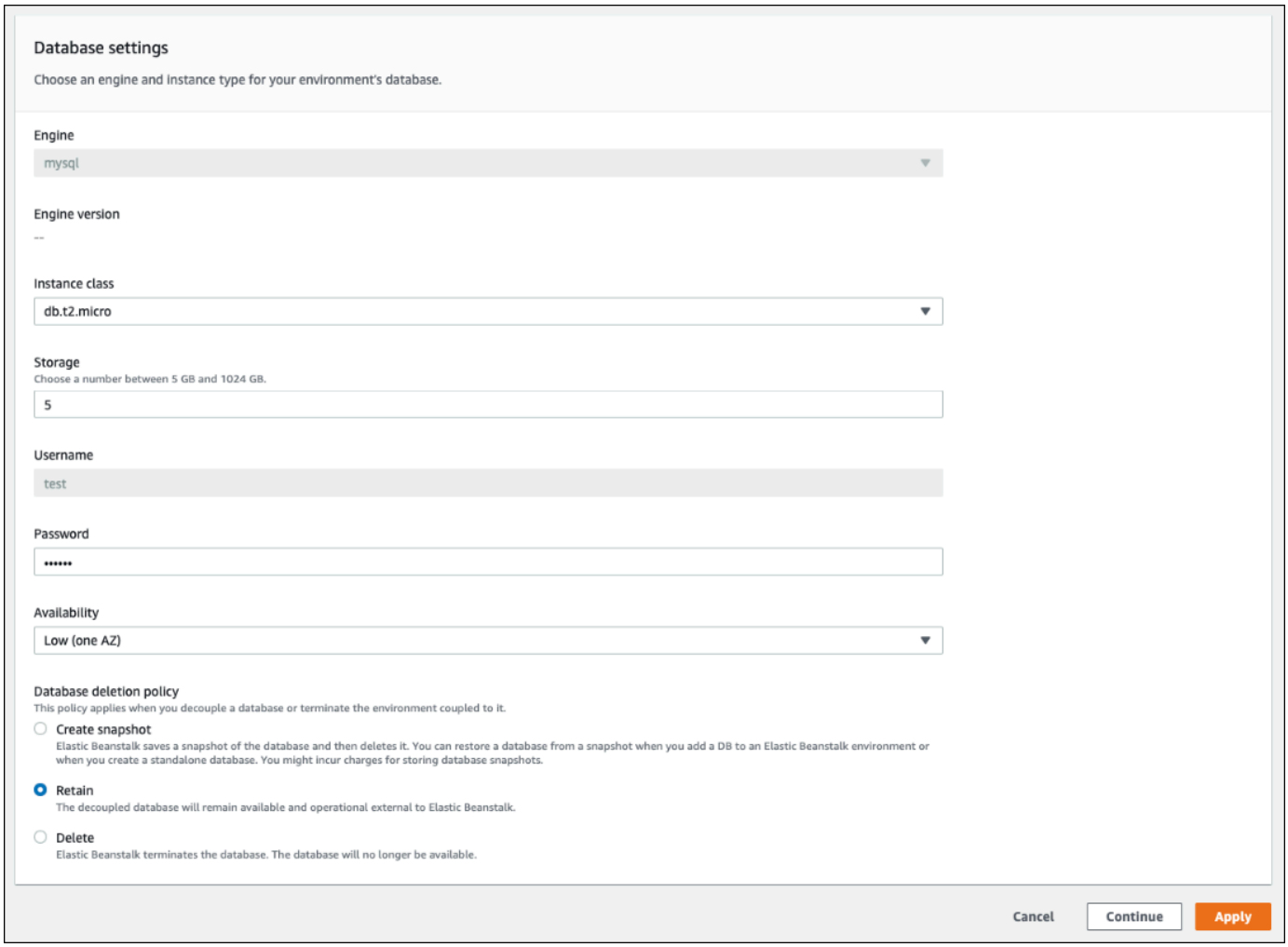
  如果您想对数据库进行任何配置更改，那么应该在数据库解耦之前应用它们。这包括对 Database deletion policy（数据库删除策略）的更改。仅应用解耦设置时，使用 Decouple database（解耦数据库）设置同时提交的任何待处理的更改将被忽略。

将数据库实例与环境解耦

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Database（数据库）配置类别中，选择 Edit（编辑）。
5. 查看 Database settings（数据库设置）部分中的所有配置值，尤其是 Database deletion policy（数据库删除策略），此策略决定了数据库在解耦后会发生什么情况。



如果所有其他配置设置均正确，请跳到步骤 6 来解耦数据库。

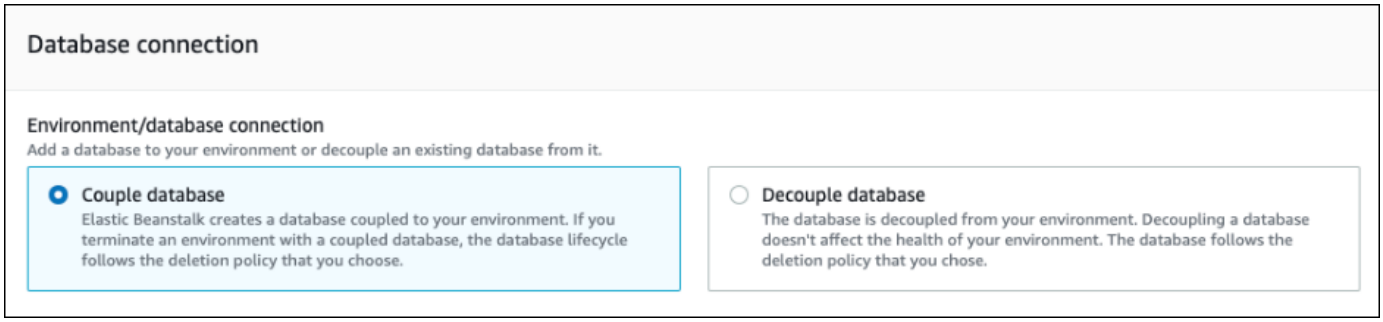
> **Warning**
>
> 将 Database deletion policy（删除数据库策略）设置与 Decouple database（解耦数据库）分开应用非常重要。如果您选择 Apply（应用）的目的是同时保存 Decouple database（解耦数据库）和新选择的 Database deletion policy（数据库删除策略），您选择的新删除策略将被忽略。Elastic Beanstalk 将按照先前设定的删除策略解耦数据库。如

果先前设置的删除策略为 Delete 或者 Create Snapshot，则会有丢失数据库的风险，而不是遵循预期的待处理策略。

如果任何配置设置需要更新，请执行以下操作：

1. 在 Database settings（数据库设置）面板中进行必要的修改。
2. 选择 Apply（应用）。保存数据库的配置更改将需要几分钟时间。
3. 返回到步骤 3，然后从导航窗格中选择 Configuration（配置）。

6. 转到窗格的 Database connection（数据库连接）部分。



7. 选择 Decouple database（解耦数据库）。
8. 选择 Apply（应用）以启动数据库解耦操作。

删除策略设置决定了数据库的结果以及解耦数据库所需的时间长度。

- 如果删除策略设置为 Delete，数据库将被删除。该操作可能需要大约 10-20 分钟，具体取决于数据库的大小。
- 如果删除策略设置为 Snapshot，将创建数据库快照。然后，数据库将被删除。此过程所需的时间长度因数据库的大小而异。
- 如果删除策略设置为 Retain，数据库在 Elastic Beanstalk 环境之外仍可运行。解耦数据库通常需要不到五分钟的时间。

如果您决定在 Elastic Beanstalk 环境之外保留数据库，则需要采取其他步骤对其进行配置。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。如果您计划为生产环境使用解耦数据库，请验证数据库使用的存储类型适合您的工作负载。有关更多信息，请参阅《Amazon RDS 用户指南》中的数据库实例存储和修改数据库实例。

使用配置文件解耦 RDS 数据库实例

您可以将数据库实例与 Elastic Beanstalk 环境解耦，而不会影响环境的运行状况。数据库实例在数据库解耦时应用的 数据库删除策略。

解耦数据库所需的两个选项都在 the section called “aws:rds:dbinstance” 命名空间中。这些原则如下所示：

- `DBDeletionPolicy` 选项设置了删除策略。可以设置为下列值之一：`Snapshot`、`Delete` 或 `Retain`。这些值在同一主题中的数据库生命周期中进行了描述。
- `HasCoupledDatabase` 选项确定您的环境是否有耦合数据库。
  - 如果切换为 `true`，Elastic Beanstalk 会创建一个与您的环境耦合的新数据库实例。
  - 如果切换为 `false`，Elastic Beanstalk 开始将数据库实例与您的环境解耦。

如果要在解耦之前更改数据库配置，请先在单独的操作中应用任何配置更改。这包括更改 `DBDeletionPolicy` 配置。应用更改后，请运行单独的命令来设置解耦选项。如果同时提交其他配置设置和解耦设置，则在应用解耦设置时忽略其他配置选项设置。

> **Warning**
>
> 请您务必运行这些命令以将 `DBDeletionPolicy` 和 `HasCoupledDatabase` 设置作为两个单独的操作应用。如果活跃删除策略已设置为 `Delete` 或者 `Snapshot`，您会有丢失数据库的风险。数据库遵循当前处于活跃状态的删除策略，而不是您预期的待处理删除策略。

将数据库实例与环境解耦

请按照以下步骤将数据库与 Elastic Beanstalk 环境解耦。您可以使用 EB CLI 或 AWS CLI 以完成这些步骤。有关更多信息，请参阅使用配置文件的高级环境自定义项。

1. 如果要更改删除策略，请按以下格式设置配置文件。在此示例中，删除策略设置为保留。

   Example

   ```
   option_settings:
     aws:rds:dbinstance:
       DBDeletionPolicy: Retain
   ```

2. 使用首选工具运行命令以完成配置更新。

3. 设置配置文件以将 HasCoupledDatabase 设置为 false。

   Example

   ```
   option_settings:
     aws:rds:dbinstance:
       HasCoupledDatabase: false
   ```

4. 使用首选工具运行命令以完成配置更新。

删除策略设置决定了数据库的结果以及解耦数据库所需的时间长度。

- 如果删除策略设置为 Delete，数据库将被删除。该操作可能需要大约 10-20 分钟，具体取决于数据库的大小。
- 如果删除策略设置为 Snapshot，将创建数据库快照。然后，数据库将被删除。此过程所需的时间长度因数据库的大小而异。
- 如果删除策略设置为 Retain，数据库在 Elastic Beanstalk 环境之外仍可运行。解耦数据库通常需要不到五分钟的时间。

如果您决定在 Elastic Beanstalk 环境之外保留数据库，则需要采取其他步骤对其进行配置。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon RDS 结合使用。如果您计划为生产环境使用解耦数据库，请验证数据库使用的存储类型适合您的工作负载。有关更多信息，请参阅《Amazon RDS 用户指南》中的数据库实例存储和修改数据库实例。

您的 AWS Elastic Beanstalk 环境安全

Elastic Beanstalk 提供了多个选项来控制您的环境及其中的 Amazon EC2 实例的服务访问（安全性）。本主题将讨论这些选项的配置。

Sections

配置环境安全性

您可以在 Elastic Beanstalk 控制台中修改 Elastic Beanstalk 环境的安全性配置。

在 Elastic Beanstalk 控制台中配置环境服务访问（安全性）

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

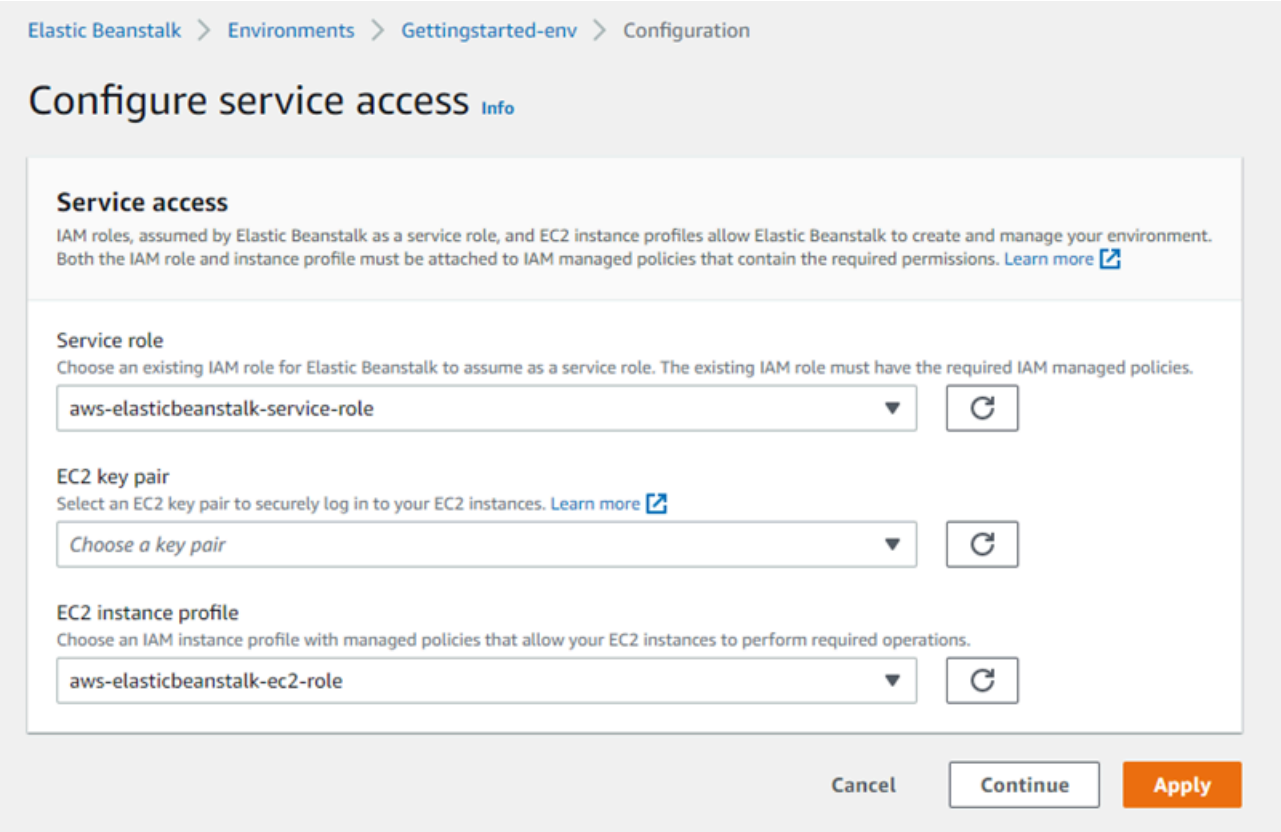
> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Service access（服务访问）配置类别中，选择 Edit（编辑）。

可供使用的设置如下。

设置

- 服务角色
- EC2 密钥对
- IAM 实例配置文件



服务角色

选择要与 Elastic Beanstalk 环境关联的服务角色。Elastic Beanstalk 在代表你 AWS 访问其他服务时担任服务角色。有关更多信息，请参阅 管理 Elastic Beanstalk 服务角色。

EC2 密钥对

您可以使用 Amazon EC2 密钥对安全地登录为 Elastic Beanstalk 应用程序预配置的 Amazon Elastic Compute Cloud (Amazon EC2) 实例。有关创建密钥对的说明，请参阅 Amazon EC2 用户指南中的使用 Amazon EC2 创建密钥对。

> **Note**
>
> 创建密钥对时，Amazon EC2 将存储公有密钥的副本。如果不再需要使用它连接到任何环境实例，则可将其从 Amazon EC2 中删除。有关详细信息，请参阅 Amazon EC2 用户指南中的删除您的密钥对。

从下拉菜单中选择 EC2 密钥对以将其分配到您环境的实例。当您分配密钥对时，公共密钥存储在实例上，用于对存储在本地的私有密钥进行身份验证。私钥永远不会存储在上 AWS。

有关连接亚马逊 EC2 实例的更多信息，请参阅 Amazon EC2 用户指南中的连接到您的实例和使用 Putty 从 Windows 连接到 Linux/UNIX 实例。

IAM 实例配置文件

EC2 实例配置文件是一种 IAM 角色，将应用到在您的 Elastic Beanstalk 环境中启动的实例。Amazon EC2 实例扮演实例配置文件角色来签署请求 AWS 并访问 API，例如，将日志上传到 Amazon S3。

首次在 Elastic Beanstalk 控制台中创建环境时，Elastic Beanstalk 提示您创建具有默认权限集的实例配置文件。您可以向此配置文件添加权限，以向您的实例提供对其他 AWS 服务的访问权限。有关更多信息，请参阅 管理 Elastic Beanstalk 实例配置文件。

> **Note**
>
> 以前，Elastic Beanstalk 创建了一个默认 EC2 实例`aws-elasticbeanstalk-ec2-role`配置文件，该配置文件名为账户首次创建环境 AWS 时命名。该实例配置文件包含默认的托管式策略。如果您的账户已经有该实例配置文件，则可继续将其分配到您的环境。
> 但是，最近的 AWS 安全准则不允许 AWS 服务自动创建具有对其他 AWS 服务（在本例中为 EC2）的信任策略的角色。根据这些安全准则，Elastic Beanstalk 将不再创建默认的 `aws-elasticbeanstalk-ec2-role` 实例配置文件。

环境安全性配置命名空间

Elastic Beanstalk 在以下命名空间中提供了配置选项来使您能够自定义环境的安全性：

- `aws:elasticbeanstalk:environment` - 使用 `ServiceRole` 选项配置环境的服务角色。
- `aws:autoscaling:launchconfiguration` - 使用 `EC2KeyName` 和 `IamInstanceProfile` 选项为环境的 Amazon EC2 实例配置权限。

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关详细信息，请参阅 建议值。

在 Elastic Beanstalk 环境中标记资源

您可以将标签应用于您的 AWS Elastic Beanstalk 环境。标签是与资源关联的键值对。 AWS 有关 Elastic Beanstalk 资源标记、使用案例、标签键和值约束以及支持的资源类型的信息，请参阅标记 Elastic Beanstalk 应用程序资源。

Elastic Beanstalk 将环境标签应用于环境资源本身，以及 Elastic Beanstalk 为环境创建的其他 AWS 资源。您可以使用标签管理某个环境内的特定资源级权限。有关更多信息，请参阅 Amazon EC2 用户指南中的为你的 Amazon EC2 资源添加标签。

默认情况下，Elastic Beanstalk 会将几个标签应用于您的环境：

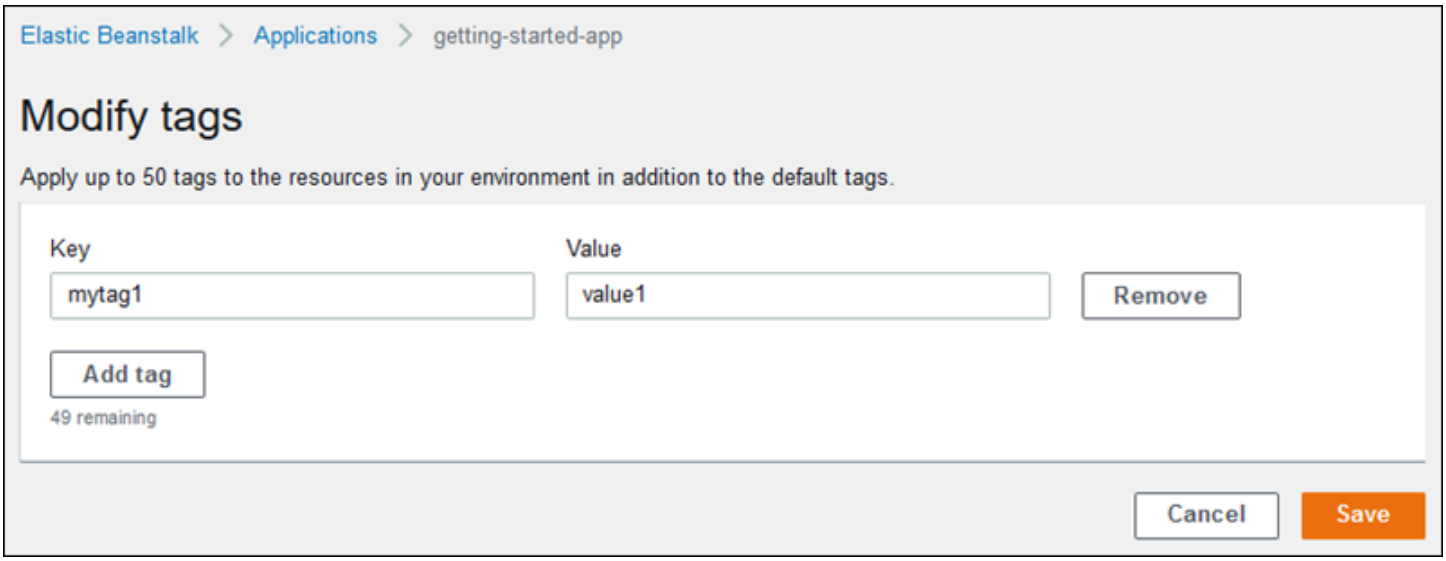
- `elasticbeanstalk:environment-name` – 环境名称。
- `elasticbeanstalk:environment-id` – 环境 ID。
- `Name` – 也是环境名称。`Name` 在 Amazon EC2 控制面板中用于对资源进行标识和排序。

您不能编辑这些默认标签。

您可以在创建 Elastic Beanstalk 环境时指定标签。在现有环境中，您可以添加或删除标签，以及更新现有标签的值。一个环境最多可以有 50 个标签，包括默认标签。

在创建环境期间添加标签

在使用 Elastic Beanstalk 控制台创建环境时，可以在创建新环境向导的修改标签配置页面上指定标签键和值。



如果使用 EB CLI 创建环境，则可以使用 eb create 的 `--tags` 选项添加标签。

```
~/workspace/my-app$ eb create --tags mytag1=value1,mytag2=value2
```

对于 AWS CLI 或其他基于 API 的客户端，请使用命令中的`--tags`参数。create-environment

```
$ aws elasticbeanstalk create-environment \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --environment-name my-env --cname-prefix my-app --
version-label v1 --template-name my-saved-config
```

保存的配置包括用户定义的标签。当您在创建环境期间应用包含标签的已保存配置时，只要您不指定任何新标签，这些标签就将应用于新环境。如果使用前述方法之一向环境添加标签，则将弃用已保存配置中定义的任何标签。

管理现有环境的标签

您可以在现有的 Elastic Beanstalk 环境中添加、更新和删除标签。Elastic Beanstalk 会将更改应用到您的环境中的资源。

不过，您不能编辑 Elastic Beanstalk 应用于您的环境的默认标签。

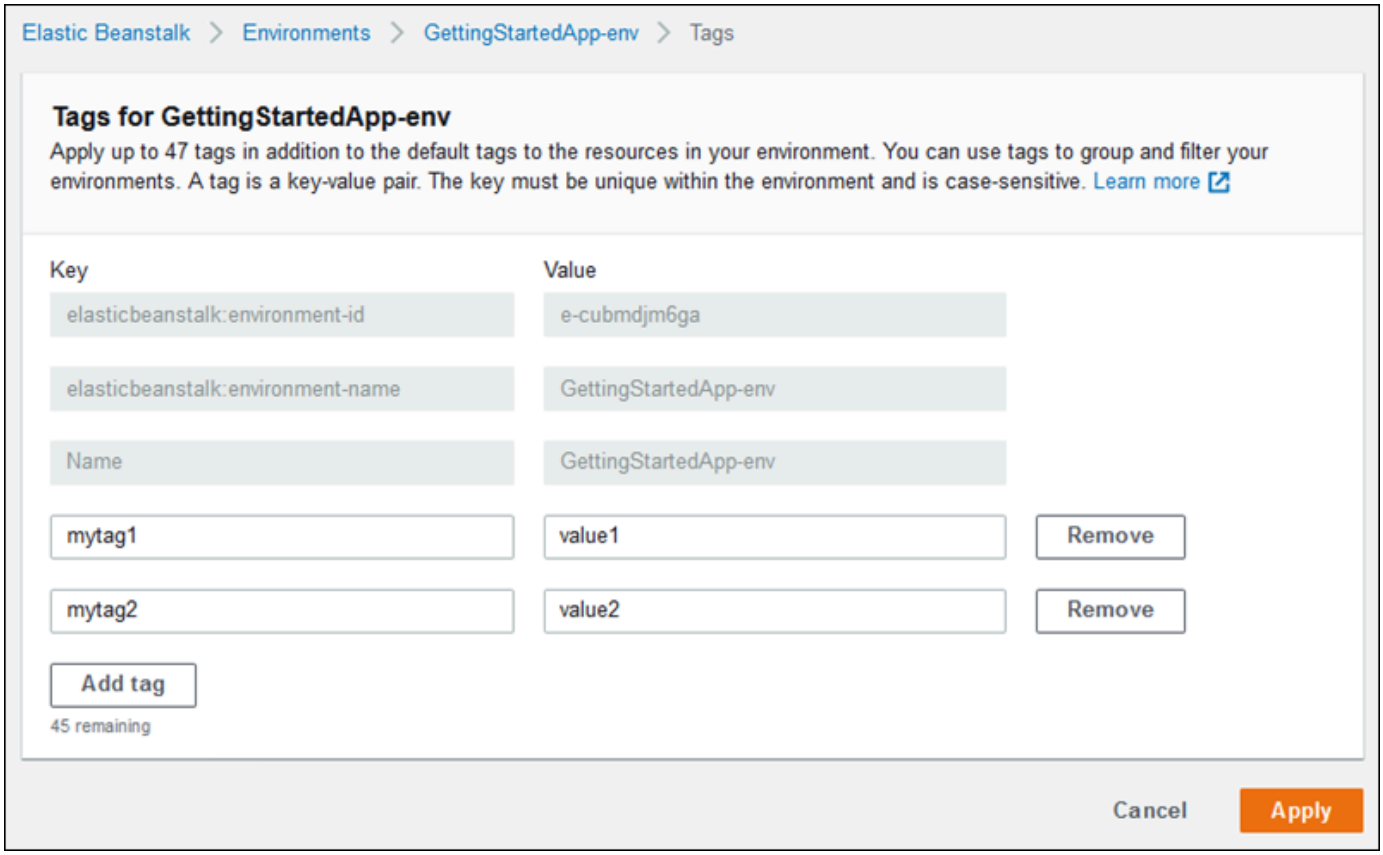
在 Elastic Beanstalk 控制台中管理环境的标签

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Tags。

   标签管理页会显示环境中当前存在的标签的列表。



4. 添加、更新或删除标签：

   - 要添加标签，请将其输入列表底部的空白框中。要添加另一个标签，请选择添加标签，Elastic Beanstalk 将添加另一对空白框。
   - 要更新标签的键或值，请编辑标签行中的相应框。
   - 要删除标签，请选择标签的值框旁边的 Remove (删除)。

5. 要保存更改，请选择页面底部的 Apply（应用）。

如果使用 EB CLI 更新环境，则可使用 eb tags 来添加、更新、删除或列出标签。

例如，以下命令会列出默认环境中的标签。

```
~/workspace/my-app$ eb tags --list
```

以下命令会更新标签 `mytag1` 并删除标签 `mytag2`。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2
```

有关选项和更多示例的完整列表，请参阅 `eb tags`。

对于 AWS CLI 或其他基于 API 的客户端，使用list-tags-for-resource命令列出环境的标签。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:environment/my-app/my-env"
```

使用 update-tags-for-resource 命令可在环境中添加、更新或删除标签。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:environment/my-
app/my-env"
```

在 `--tags-to-add` 的 update-tags-for-resource 参数中指定要添加的标签和要更新的标签。添加了一个不存在的标签，更新了现有标签的值。

> **Note**
>
> 要在 Elastic Beanstalk 环境中使用这两个 AWS CLI 命令，你需要环境的 ARN。您可以使用下面的命令检索该 ARN。
>
> ```
> $ aws elasticbeanstalk describe-environments
> ```

环境属性和其他软件设置

通过配置更新、监控和日志记录配置页面，您可以在运行应用程序的 Amazon Elastic Compute Cloud（Amazon EC2）实例上配置软件。您可以配置环境属性、AWS X-Ray 调试、实例日志的存储和流式传输以及特定于平台的设置。

主题

- 配置特定于平台的设置
- 配置环境属性（环境变量）
- 软件设置命名空间
- 访问环境属性
- 配置 AWS X-Ray 调试
- 查看您的 Elastic Beanstalk 环境日志

配置特定于平台的设置

除了可用于所有环境的标准选项集之外，大多数 Elastic Beanstalk 平台还允许您指定特定于语言或框架的设置。这些设置显示在配置更新、监控和日志记录页面的平台软件部分中，并且可以采用以下形式：

- 预设环境属性 - Ruby 平台将环境属性用于框架设置，例如 `RACK_ENV` 和 `BUNDLE_WITHOUT`。
- 占位符环境属性 - Tomcat 平台定义名为 `JDBC_CONNECTION_STRING` 且未设置为任何值的环境属性。此类设置在较旧的平台版本中更常见。
- 配置选项 - 大多数平台在特定于平台或共享的命名空间（如 `aws:elasticbeanstalk:xray` 或 `aws:elasticbeanstalk:container:python`）中定义配置选项。

在 Elastic Beanstalk 控制台中配置特定于平台的设置

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在平台软件下，进行必要的选项设置更改。
6. 要保存更改，请选择页面底部的 Apply（应用）。

有关特定于平台的选项以及有关在代码中获取环境属性值的信息，请参阅适用于您的语言或框架的平台主题：

- Docker – the section called “环境配置”
- Go – 使用 Elastic Beanstalk Go 平台
- Java SE – 使用 Elastic Beanstalk Java SE 平台
- Tomcat – 使用 Elastic Beanstalk Tomcat 平台
- .NET Core on Linux – 使用 .NET Core on Linux 平台
- .NET – 使用 Elastic Beanstalk .NET 平台

- Node.js – 使用 Elastic Beanstalk Node.js 平台
- PHP – 使用 Elastic Beanstalk PHP 平台
- Python – 使用 Elastic Beanstalk Python 平台
- Ruby – 使用 Elastic Beanstalk Ruby 平台

配置环境属性（环境变量）

您可以使用环境属性（也称为环境变量）向应用程序传递密钥、端点、调试设置和其他信息。环境属性可以帮助您在多个环境中为不同目的运行应用程序，如开发、测试、暂存和生产。

此外，当您向环境中添加数据库时，Elastic Beanstalk 会设置环境属性（例如 RDS_HOSTNAME），您可以在应用程序代码中读取这些属性来构建连接对象或字符串。

> **环境变量**
>
> 大多数情况下，环境属性作为环境变量 传递到应用程序，但行为因平台而异。例如，Java SE 平台设置您用 `System.getenv` 检索的环境变量，而 Tomcat 平台则设置您用 `System.getProperty` 检索的 Java 系统属性。一般来说，如果您连接到实例并运行 `env`，则属性不会 显示出来。

在 Elastic Beanstalk 控制台中配置环境属性

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 向下滚动到环境属性。
6. 选择添加环境属性。
7. 输入属性名称和值对。
8. 如需添加更多变量，请重复步骤 6 和步骤 7。

9. 要保存更改，请选择页面底部的 Apply（应用）。

环境属性限制

- 键可以包含任意字母数字字符和以下符号：`_ . : / + \ - @`

  列出的符号对环境属性键有效，但可能对您的环境平台上的环境变量名称无效。为了与所有平台兼容，请将环境属性限制于以下模式：`[A-Z_][A-Z0-9_]*`
- 值可以包含任意字母数字字符、空格和以下符号：`_ . : / = + \ - @ ' "`

> **Note**
>
> 环境属性值中的一些字符必须进行转义。可使用反斜杠字符 (\) 表示一些特殊字符和控制字符。以下列表包含表示需要转义的一些字符的示例：
>
> - 反斜杠 (\) — 表示使用 `\\`
> - 单引号 (') — 表示使用 `\'`
> - 双引号 (") — 表示使用 `\"`

- 键和值区分大小写。
- 当以#=# 的格式存储为字符串时，所有环境属性的组合大小不得超过 4096 字节。

软件设置命名空间

您可以使用配置文件设置配置选项并在部署期间执行其他实例配置。配置选项可以通过您使用的 Elastic Beanstalk 服务或平台定义并组织到命名空间 中。

您可以使用 Elastic Beanstalk 配置文件在源代码中设置环境属性和配置选项。使用 `aws:elasticbeanstalk:application:environment` 命名空间定义环境属性。

Example .ebextensions/options.config

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    API_ENDPOINT: www.example.com/api
```

如果您使用配置文件或 AWS CloudFormation 模板创建自定义资源，可以使用 AWS CloudFormation 函数来获取有关资源的信息并在部署期间将其动态分配给环境属性。来自 elastic-beanstalk-samples

GitHub 存储库的以下示例使用 Ref 函数获取它所创建的 Amazon SNS 主题的 ARN，并将其分配给名为 NOTIFICATION_TOPIC 的环境属性。

> **注意**
>
> - 如果您使用 AWS CloudFormation 函数定义环境属性，则 Elastic Beanstalk 控制台会在计算该函数之前显示属性的值。您可以使用 get-config 平台脚本 来确定适用于您的应用程序的环境属性的值。
> - 多容器 Docker 平台不使用 AWS CloudFormation 创建容器资源。因此，此平台不支持使用 AWS CloudFormation 函数定义环境属性。

Example .Ebextensions/sns-topic.config

```
Resources:
  NotificationTopic:
    Type: AWS::SNS::Topic

option_settings:
  aws:elasticbeanstalk:application:environment:
    NOTIFICATION_TOPIC: ``{"Ref" : "NotificationTopic"}``
```

您也可以使用此功能从 AWS CloudFormation 虚拟参数传播信息。此示例获取当前区域并将其分配给名为 AWS_REGION 的属性。

Example .Ebextensions/env-regionname.config

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    AWS_REGION: ``{"Ref" : "AWS::Region"}``
```

大多数 Elastic Beanstalk 平台都使用用于配置在实例中运行的软件的选项定义其他命名空间，如可将请求中继到您的应用程序的反向代理。有关可用于您的平台的命名空间的更多信息，请参阅以下内容：

- Go – Go 配置命名空间
- Java SE – Java SE 配置命名空间
- Tomcat – Tomcat 配置命名空间
- .NET Core on Linux – .NET Core on Linux 配置命名空间

- .NET – aws:elasticbeanstalk:container:dotnet:apppool 命名空间
- Node.js – Node.js 配置命名空间
- PHP – aws:elasticbeanstalk:container:php:phpini 命名空间
- Python – Python 配置命名空间
- Ruby – Ruby 配置命名空间

Elastic Beanstalk 提供了许多用于自定义环境的配置选项。除了配置文件之外，您还可使用控制台、保存的配置、EB CLI 或 AWS CLI 来配置选项。参阅 配置选项 了解更多信息。

访问环境属性

大多数情况下，您在应用程序代码 (如环境变量) 中访问环境属性。但是，环境属性通常只传递给应用程序，不能通过在您的环境中连接实例和运行 env 来查看。

- Go – `os.Getenv`

```
endpoint := os.Getenv("API_ENDPOINT")
```

- Java SE – `System.getenv`

```
String endpoint = System.getenv("API_ENDPOINT");
```

- Tomcat – `System.getProperty`

```
String endpoint = System.getProperty("API_ENDPOINT");
```

- .NET Core on Linux – `Environment.GetEnvironmentVariable`

```
string endpoint = Environment.GetEnvironmentVariable("API_ENDPOINT");
```

- .NET – `appConfig`

```
NameValueCollection appConfig = ConfigurationManager.AppSettings;
string endpoint = appConfig["API_ENDPOINT"];
```

- Node.js – `process.env`

```
var endpoint = process.env.API_ENDPOINT
```

- PHP – `$_SERVER`

```
$endpoint = $_SERVER['API_ENDPOINT'];
```

- Python – `os.environ`

```
import os
endpoint = os.environ['API_ENDPOINT']
```

- Ruby – ENV

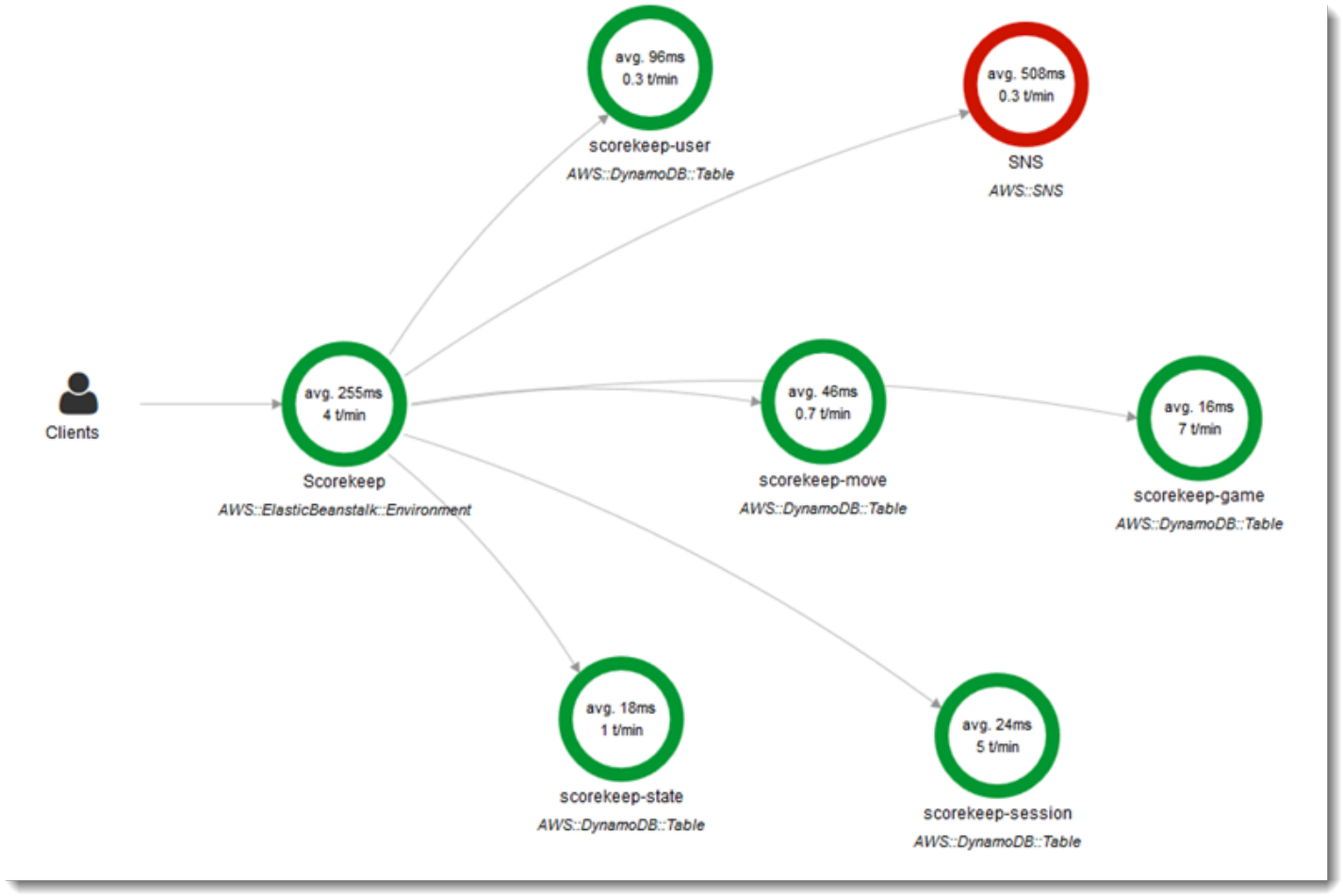
```
endpoint = ENV['API_ENDPOINT']
```

除了应用程序代码（如在部署过程中运行的脚本）外，您还可以使用`get-config` 平台脚本来访问环境属性。请参阅 elastic-beanstalk-samples GitHub 存储库中使用 `get-config` 的示例配置。

配置 AWS X-Ray 调试

您可以使用 AWS Elastic Beanstalk 控制台或配置文件在环境中的实例上运行 AWS X-Ray 守护程序。X-Ray 是一项AWS服务，用于收集有关应用程序服务的请求的数据，并使用它来构建服务地图，以便您发现应用程序问题和优化机会。

> **Note**
>
> 有些区域不提供 X-Ray。如果您在其中一个区域中创建环境，则不能在环境中的实例上运行 X-Ray 守护程序。
> 有关每个区域中提供的 AWS 服务的信息，请参阅区域表。



X-Ray 提供了一个可用于检测应用程序代码的开发工具包，以及一个用于将调试信息从开发工具包中继到 X-Ray API 的守护程序。

支持的平台

您可以将 X-Ray 开发工具包与以下 Elastic Beanstalk 平台结合使用：

- Go - 版本 2.9.1 及更高版本
- Java 8 - 版本 2.3.0 及更高版本
- Java 8 with Tomcat 8 - 版本 2.4.0 及更高版本
- Node.js - 版本 3.2.0 及更高版本
- Windows Server - 在 2016 年 12 月 18 日或之后发布的所有平台版本
- Python - 版本 2.5.0 及更高版本

在支持的平台上，您可以使用配置选项在环境中的实例上运行 X-Ray 守护程序。您可以在 Elastic Beanstalk 控制台中或使用配置文件启用此守护程序。

为了将数据上传到 X-Ray，X-Ray 守护程序需要 AWSXrayWriteOnlyAccess 托管策略中的 IAM 权限。这些权限包含在 Elastic Beanstalk 实例配置文件中。如果您不使用默认的实例配置文件，请参阅 AWS X-Ray 开发人员指南中的为守护进程授予向 X-Ray 发送数据的权限。

要使用 X-Ray 进行调试，必须使用 X-Ray 开发工具包。有关说明和示例应用程序，请参阅 AWS X-Ray 开发人员指南中的开始使用 AWS X-Ray。

如果使用不包含守护程序的平台版本，您仍然可以在配置文件中使用脚本运行它。有关更多信息，请参阅 AWS X-Ray 开发人员指南中的手动下载和运行 X-Ray 守护进程（高级）。

小节目录

配置调试

您可以通过 Elastic Beanstalk 控制台在运行环境中启用 X-Ray 守护程序。

在 Elastic Beanstalk 控制台中启用调试

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在 Amazon X-Ray 部分中，选择启用。
6. 要保存更改，请选择页面底部的 Apply（应用）。

您也可以在创建环境期间启用此选项。有关更多信息，请参阅创建新环境向导。

aws:elasticbeanstalk:xray 命名空间

您可以使用 XRayEnabled 命名空间中的 aws:elasticbeanstalk:xray 选项启用调试。

要在部署应用程序时自动启用调试，请在源代码中的配置文件中设置此选项，如下所示。

Example .ebextensions/debugging.config

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

查看您的 Elastic Beanstalk 环境日志

AWS Elastic Beanstalk 提供了两种方式来从运行应用程序的 Amazon EC2 实例定期查看日志：

- 配置您的 Elastic Beanstalk 环境以将轮换的实例日志上传到环境的 Amazon S3 存储桶。
- 配置环境以将实例日志流式传输到 Amazon CloudWatch Logs。

当您配置到 CloudWatch Logs 的实例日志流式传输时，Elastic Beanstalk 会在 Amazon EC2 实例上为代理和部署日志创建 CloudWatch Logs 日志组，并将这些日志文件实时传输到 CloudWatch Logs。有关实例日志的更多信息，请参阅查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志。

除了实例日志之外，如果您为环境启用增强型运行状况，则可以将环境配置为将运行状况信息流式传输至 CloudWatch Logs。当环境的运行状况状态发生更改时，Elastic Beanstalk 会将记录连同新状态和更改原因说明一起添加到运行状况日志组。有关环境运行状况流式传输的信息，请参阅将 Elastic Beanstalk 环境运行状况信息流式传输到 Amazon CloudWatch Logs。

配置实例日志查看

要查看实例日志，您可以在 Elastic Beanstalk 控制台中启用实例日志轮换和日志流式传输。

在 Elastic Beanstalk 控制台中配置实例日志轮换和日志流式传输

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在 S3 日志存储部分中，选择轮换日志下方的启用以允许将轮换日志上传到 Amazon S3。
6. 在 Instance log streaming to CloudWatch Logs (实例日志流式传输到 CloudWatch Logs) 部分中，配置以下设置：
   - 日志流式传输：选择启用以启用日志流式传输。
   - Retention (保留) - 指定在 CloudWatch Logs 中保留日志的天数。
   - Lifecycle (生命周期) - 设置为 Delete logs upon termination (终止时删除日志)，以便在环境终止时立即从 CloudWatch Logs 中删除日志，而不是等待日志到期。
7. 要保存更改，请选择页面底部的 Apply（应用）。

在启用日志流式传输后，您可以返回到软件配置类别或页面并查找日志组链接。单击此链接可在 CloudWatch 控制台中查看您的实例日志。

配置环境运行状况日志查看

要查看环境运行状况日志，您可以在 Elastic Beanstalk 控制台中启用环境运行状况日志流式传输。

在 Elastic Beanstalk 控制台中配置环境运行状况日志流式传输

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 转到监控部分。
6. 在 Health event streaming to CloudWatch Logs (运行状况事件流式传输到 CloudWatch Logs) 下，配置以下设置：
   - 日志流式传输：选择启用以启用日志流式传输。
   - Retention (保留) - 指定在 CloudWatch Logs 中保留日志的天数。

- Lifecycle (生命周期) - 设置为 Delete logs upon termination (终止时删除日志)，以便在环境终止时立即从 CloudWatch Logs 中删除日志，而不是等待日志到期。

7. 要保存更改，请选择页面底部的 Apply（应用）。

日志查看命名空间

以下命名空间包含日志查看的设置：

- `aws:elasticbeanstalk:hostmanager` - 配置为将轮换日志上传到 Amazon S3。
- `aws:elasticbeanstalk:cloudwatch:logs` - 配置为将实例日志流式传输到 CloudWatch。
- `aws:elasticbeanstalk:cloudwatch:logs:health` - 配置为将环境运行状况流式传输到 CloudWatch。

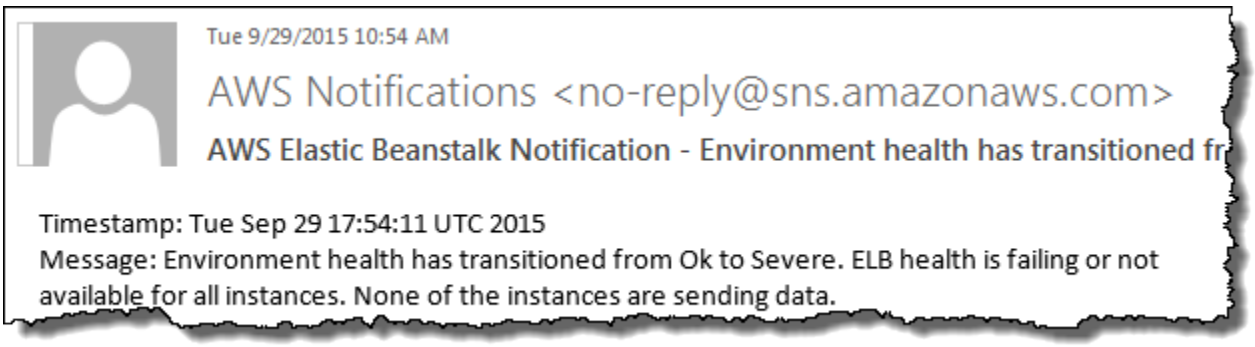
使用 Amazon SNS 发送 Elastic Beanstalk 环境通知

您可以配置 AWS Elastic Beanstalk 环境，以使用 Amazon Simple Notification Service (Amazon SNS) 向您通知影响应用程序的重要事件。要在发生错误或环境运行状况发生变化时接收来自AWS的电子邮件，请在创建环境时或随后指定电子邮件地址。

> **Note**
>
> Elastic Beanstalk 使用 Amazon SNS 进行通知。有关 Amazon SNS 定价的信息，请参阅 https://aws.amazon.com/sns/pricing/。

为环境配置通知时，Elastic Beanstalk 会代表您为环境创建 Amazon SNS 主题。Elastic Beanstalk 必须具有所需的权限才能发送消息到 Amazon SNS 主题。有关更多信息，请参阅配置发送通知的权限。

当值得注意的事件发生时，Elastic Beanstalk 会向该主题发送消息。然后，Amazon SNS 则将它收到的消息中继到该主题的订阅者。值得注意的事件包括环境创建错误以及环境和实例运行状况的所有更改。针对 Amazon EC2 Auto Scaling 操作（例如，对环境添加和删除实例）的事件和其他信息性事件不会触发通知。



您可以在创建环境时或此后的某个时间在 Elastic Beanstalk 控制台中输入电子邮件地址。这样将会创建一个 Amazon SNS 主题并订阅该主题。Elastic Beanstalk 负责管理主题的生命周期，在您的环境终止时或者您在环境管理控制台中删除电子邮件地址时删除主题。

`aws:elasticbeanstalk:sns:topics` 命名空间提供了各种选项，可用于使用配置文件、CLI 或软件开发工具包来配置 Amazon SNS 主题。使用这些方法之一，您可以配置订阅者和终端节点的类型。对于订阅者类型，您可以选择 Amazon SQS 队列或 HTTP URL。

您只能打开或关闭 Amazon SNS 通知。根据环境的大小和构成，发送到主题的通知频率可能较高。要配置在特定情况下发送的通知，您还可以使用其他选项。您可以使用 Amazon EventBridge 设置事件驱动型规则，它可在 Elastic Beanstalk 发出符合特定标准的事件时通知您。或者，您还可以配置环境以发布自定义指标，并设置 Amazon CloudWatch 警报，以在这些指标达到运行状况不佳阈值时通知您。

使用 Elastic Beanstalk 控制台配置通知

您可以在 Elastic Beanstalk 控制台中输入电子邮件地址，以便为环境创建 Amazon SNS 主题。

在 Elastic Beanstalk 控制台中配置通知

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 向下滚动到电子邮件通知部分。

6. 输入电子邮箱地址。
7. 要保存更改，请选择页面底部的 Apply（应用）。

当您为通知输入电子邮件地址时，Elastic Beanstalk 会为环境创建 Amazon SNS 主题并添加订阅。Amazon SNS 会向订阅的地址发送电子邮件以确认订阅。您必须单击确认电子邮件中的链接才能激活订阅并接收通知。

使用配置选项配置通知

可以使用 `aws:elasticbeanstalk:sns:topics` 命名空间中的选项为您的环境配置 Amazon SNS 通知。可以使用配置文件、CLI 或开发工具包来设置这些选项。

- 通知终端节点 – 要将通知发送到的电子邮件地址、Amazon SQS 队列或 URL。如果您设置此选项，则可为指定终端节点创建 SQS 队列和订阅。如果终端节点不是电子邮件地址，还必须设置 `Notification Protocol` 选项。SNS 会基于 `Notification Endpoint` 的值来验证 `Notification Protocol` 的值。多次设置此选项可创建针对主题的更多订阅。如果您清除此选项，则该主题将被删除。
- 通知协议 – 用于将通知发送到 `Notification Endpoint` 的协议。此选项默认为 `email`。将此选项设置为 `email-json` 可发送 JSON 格式的电子邮件，设置为 `http` 或 `https` 可将 JSON 格式的通知发布到 HTTP 终端节点，或设置为 `sqs` 可向 SQS 队列发送通知。

  > **Note**
  >
  > 不支持 AWS Lambda 通知。

- 通知主题 ARN – 为环境设置通知终端节点之后，读取此设置以获取 SNS 主题的 ARN。您还可以设置此选项为通知使用现有 SNS 主题。更改此选项或终止环境时，不会删除通过此选项附加到环境的主题。

  要配置 Amazon SNS 通知，您需要具备所需的权限。如果您的 IAM 用户使用 AdministratorAccess-AWSElasticBeanstalk 托管用户策略，则您应当已拥有配置 Elastic Beanstalk 为您的环境创建的默认 Amazon SNS 主题所需的权限。但是，当您配置 Elastic Beanstalk 无法管理的 Amazon SNS 主题时，则需要将以下策略添加到您的用户角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "sns:SetTopicAttributes",
        "sns:GetTopicAttributes",
        "sns:Subscribe",
        "sns:Unsubscribe",
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-2:123456789012:sns_topic_name"
      ]
    }
  ]
}
```

- 通知主题名称 – 设置此选项可自定义用于环境通知的 Amazon SNS 主题的名称。如果已存在同名主题，则 Elastic Beanstalk 会将该主题挂载到环境。

  > **Warning**
  >
  > 如果您使用 `Notification Topic Name` 将现有 SNS 主题挂载到环境，则将来您终止环境或更改此设置时，Elastic Beanstalk 将会删除该主题。

  如果您更改此选项，`Notification Topic ARN` 也会更改。如果环境已经挂载了主题，则 Elastic Beanstalk 会删除旧主题，并创建新的主题和订阅。

  使用自定义主题名称，您还必须提供外部创建的自定义主题的 ARN。托管用户策略不会自动检测使用自定义名称的主题，因此您必须向 IAM 用户提供自定义 Amazon SNS 权限。使用的策略与用于自定义主题 ARN 的策略相似，但增加以下内容：

  - `Actions` 列表中包括另外两个操作，具体是：`sns:CreateTopic`、`sns:DeleteTopic`
  - 如果要将 `Notification Topic Name` 从一个自定义主题名称更改为另一个自定义主题名称，您还必须在 `Resource` 列表中包含两个主题的 ARN。或者，包含一个涵盖两个主题的正则表达式。这样，Elastic Beanstalk 就有权删除旧主题并创建新主题。

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

配置发送通知的权限

本部分讨论与使用 Amazon SNS 的通知相关的安全注意事项。有两种截然不同的情况：

- 使用 Elastic Beanstalk 为您的环境创建的默认 Amazon SNS 主题。
- 通过配置选项提供外部 Amazon SNS 主题。

Amazon SNS 主题的默认访问策略仅允许主题所有者发布或订阅该主题。但是，通过适当的策略配置，Elastic Beanstalk 可以获得在本节中描述的两种情况之一中向 Amazon SNS 主题发布的权限。以下各子节提供了详细信息。

默认主题的权限

为环境配置通知时，Elastic Beanstalk 会为环境创建 Amazon SNS 主题。Elastic Beanstalk 必须具有所需的权限才能发送消息到 Amazon SNS 主题。如果您的环境使用 Elastic Beanstalk 控制台或 EB CLI 为其生成的服务角色，或者使用您的账户的监控服务相关角色，则无需执行任何其他操作。这些托管角色包括允许 Elastic Beanstalk 向 Amazon SNS 主题发送消息所需的权限。

但是，如果您在创建环境时提供自定义服务角色，请确保此自定义服务角色包括以下策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-2:123456789012:ElasticBeanstalkNotifications*"
      ]
    }
  ]
}
```

外部主题的权限

使用配置选项配置通知 解释了如何使用其他 Amazon SNS 主题替换 Elastic Beanstalk 提供的 Amazon SNS 主题。如果您替换了主题，Elastic Beanstalk 必须验证您有权发布到此 SNS 主题，您才能

将 SNS 主题与环境关联。您应该具有 `sns:Publish`。服务角色使用相同的权限。为验证这一点，Elastic Beanstalk 在您创建或更新环境的操作中，发送测试通知到 SNS。如果此测试失败，则创建或更新环境的尝试也会失败。Elastic Beanstalk 会显示一条消息，解释失败的原因。

如果您为环境提供了自定义服务角色，请确保自定义服务角色包括以下策略，以允许 Elastic Beanstalk 向 Amazon SNS 主题发送消息。在以下代码中，将 *sns_topic_name* 替换为您在配置选项中提供的 Amazon SNS 主题的名称。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:us-east-2:123456789012:sns_topic_name"
      ]
    }
  ]
}
```

有关 Amazon SNS 访问控制的详细信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的 Amazon SNS 访问控制示例案例。

使用 Elastic Beanstalk 配置 Amazon Virtual Private Cloud (Amazon VPC)

Amazon Virtual Private Cloud (Amazon VPC) 是网络服务，它将流量安全地路由到在 Elastic Beanstalk 中运行应用程序的 EC2 实例。如果在启动您的环境时未配置 VPC，Elastic Beanstalk 将使用默认 VPC。

您可以在自定义 VPC 中启动您的环境以自定义网络和安全设置。您可以在 Elastic Beanstalk 中选择用于您的资源的子网，以及如何为您的环境中的实例和负载均衡器配置 IP 地址。在创建环境时，它将锁定到一个 VPC，但您可以在运行的环境上更改子网和 IP 地址设置。

> **Note**
>
> 如果您在 2013 年 12 月 4 日之前创建了 AWS 账户，则环境可能在某些 AWS 区域中使用 Amazon EC2-Classic 网络配置，而不是使用 Amazon VPC。有关将环境从 EC2-Classic 迁移到 VPC 网络配置的信息，请参阅将 Elastic Beanstalk 环境从 EC2-Classic 迁移到 VPC。

在 Elastic Beanstalk 控制台中配置 VPC 设置

如果在创建环境时选择自定义 VPC，您可以在 Elastic Beanstalk 控制台中修改其 VPC 设置。

配置您的环境的 VPC 设置

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Network（网络）配置类别中，选择 Edit（编辑）。

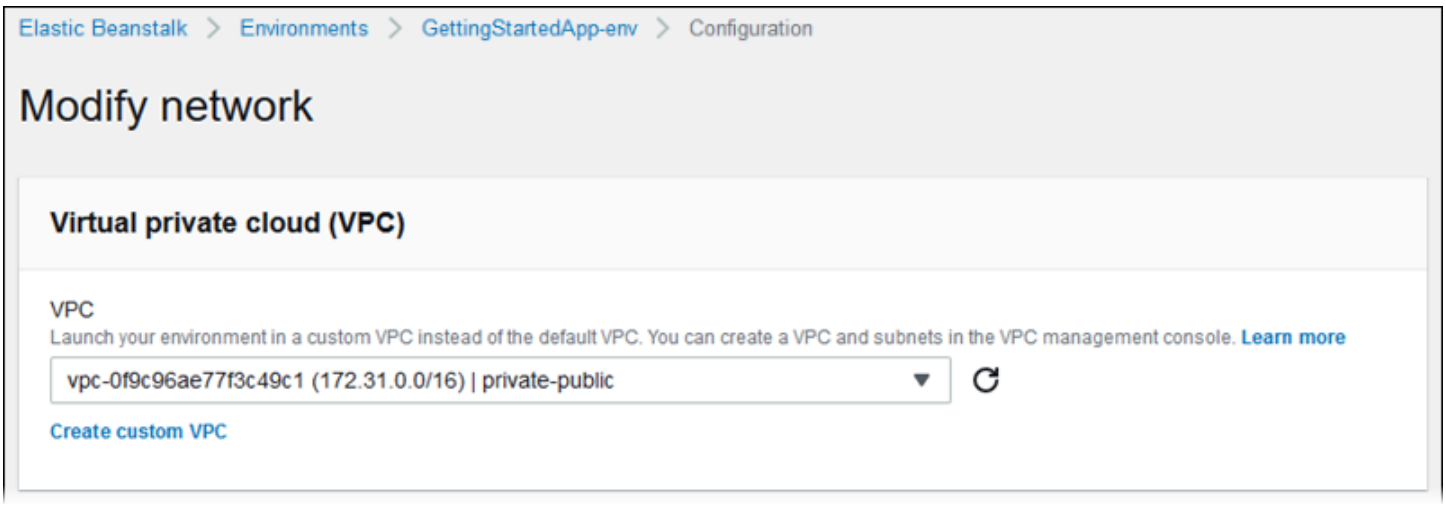
可供使用的设置如下。

选项

- VPC
- 负载均衡器可见性
- 负载均衡器子网
- 实例公有 IP 地址
- 实例子网
- 数据库子网

VPC

为您的环境选择一个 VPC。您只能在创建环境期间更改该设置。

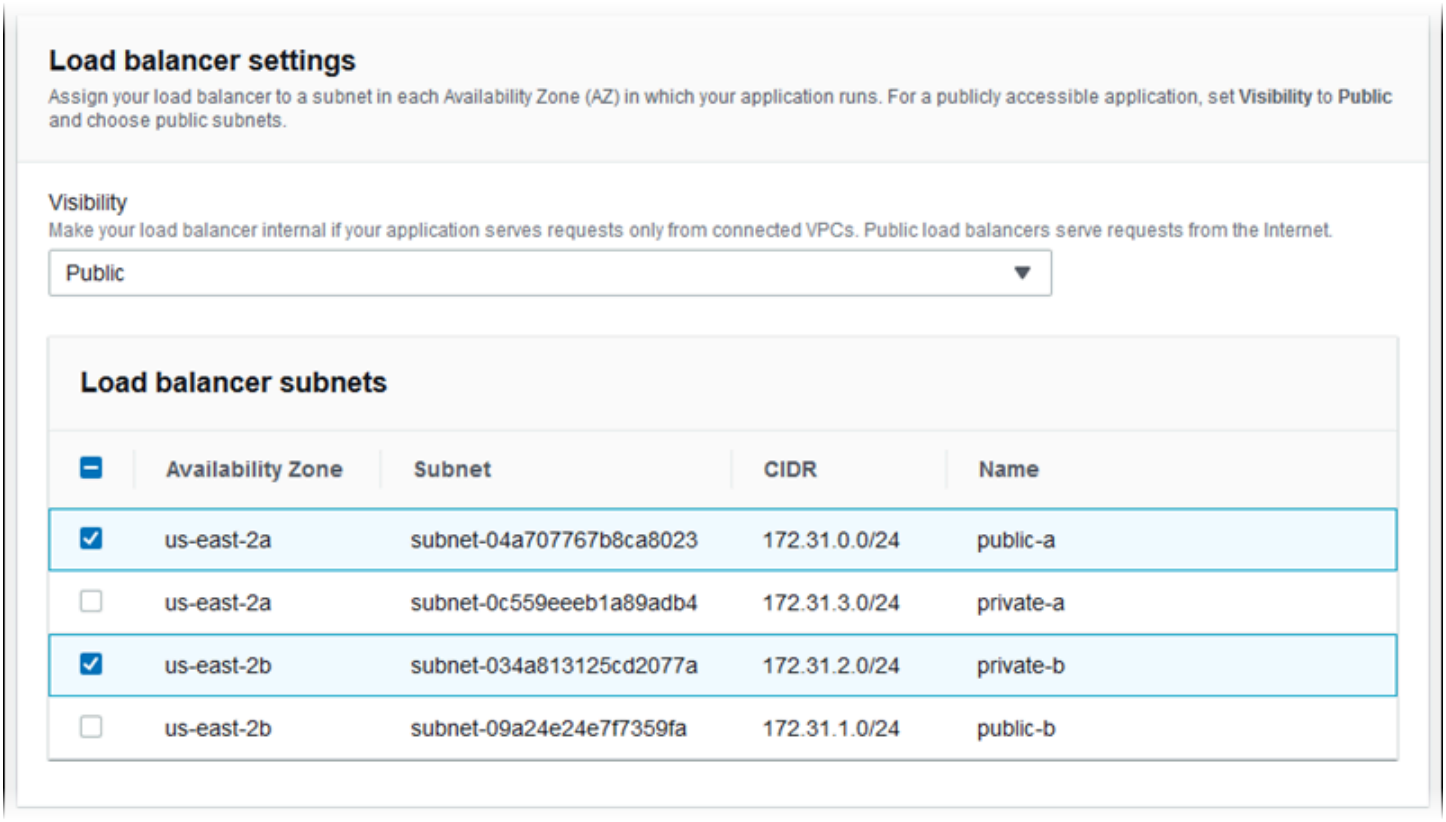


负载均衡器可见性

对于负载均衡的环境，请选择负载均衡器模式。默认情况下，负载均衡器是具有公有 IP 地址和域名的公有负载均衡器。如果您的应用程序仅处理来自您的 VPC 或连接的 VPN 的流量，请取消选择该选项，然后为您的负载均衡器选择私有子网以将负载均衡器指定为内部负载均衡器并禁止从 Internet 访问。

负载均衡器子网

对于负载均衡的环境，请选择负载均衡器用于处理流量的子网。对于公有应用程序，请选择公有子网。可以在多个可用区中使用子网以提供高可用性。对于内部应用程序，请选择私有子网并禁用负载均衡器可见性。

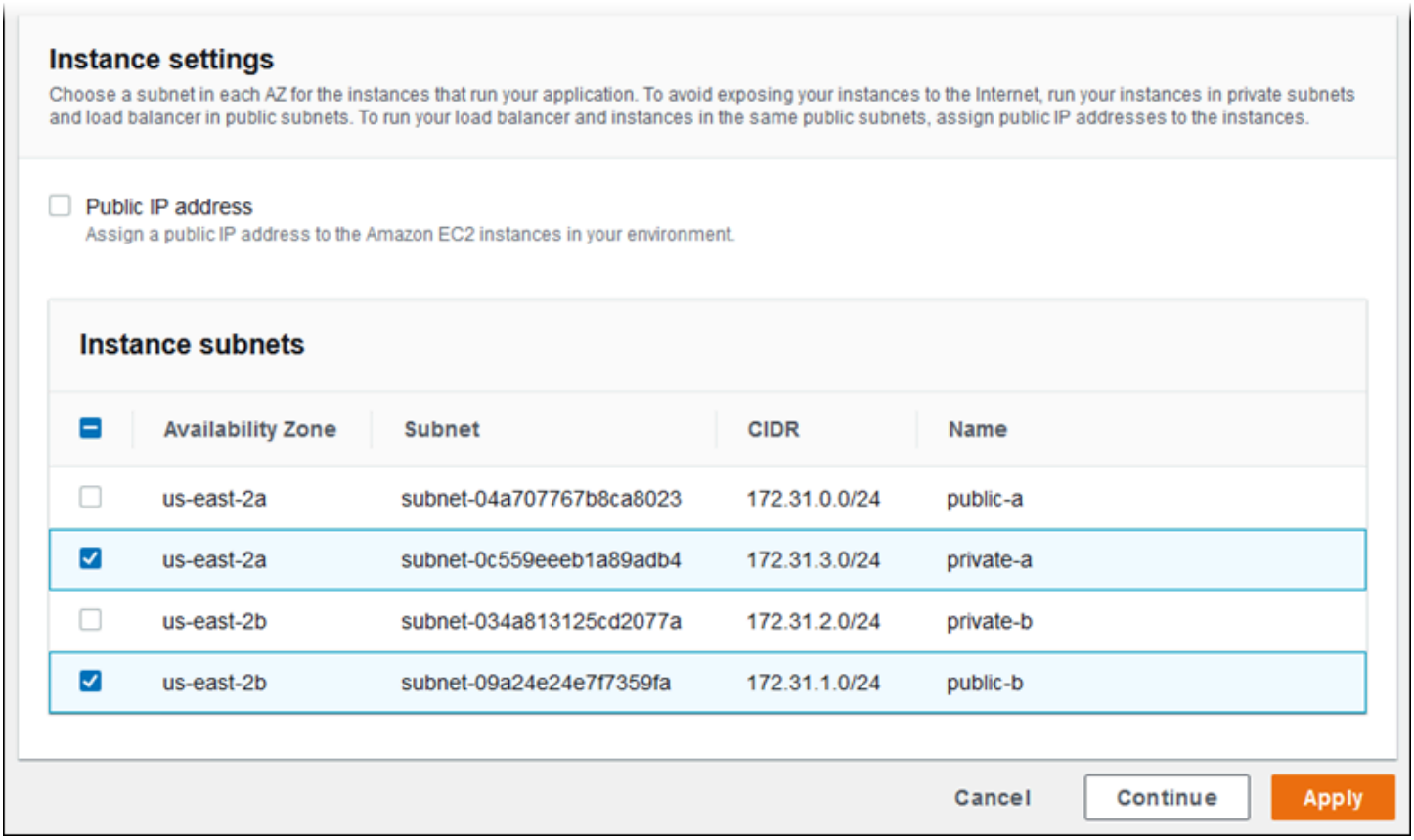


实例公有 IP 地址

如果为您的应用程序实例选择公有子网，请启用公有 IP 地址以便从 Internet 路由这些地址。

实例子网

为您的应用程序实例选择子网。为您的负载均衡器使用的每个可用区选择至少一个子网。如果为您的实例选择私有子网，您的 VPC 必须在公有子网中具有实例可用于访问 Internet 的 NAT 网关。



数据库子网

在运行附加到您的 Elastic Beanstalk 环境的 Amazon RDS 数据库时，请为数据库实例选择子网。为了获得高可用性，请将数据库指定为多可用区数据库，并为每个可用区选择一个子网。要确保您的应用程序可以连接到您的数据库，请在同一子网中运行应用程序和数据库。

aws:ec2:vpc 命名空间

您可以使用 `aws:ec2:vpc` 命名空间中的配置选项配置您的环境的网络设置。

以下配置文件使用该命名空间中的选项，为公有-私有配置设置环境的 VPC 和子网。要在配置文件中设置 VPC ID，必须在创建环境期间在应用程序源包中包含该文件。有关在创建环境期间配置这些设置的其他方法，请参阅在环境创建期间设置配置选项。

Example .ebextensions/vpc.config – 公有-私有

```
option_settings:
   aws:ec2:vpc:
      VPCId: vpc-087a68c03b9c50c84
      AssociatePublicIpAddress: 'false'
```

```
      ELBScheme: public
      ELBSubnets: subnet-0fe6b36bcb0ffc462,subnet-032fe3068297ac5b2
      Subnets: subnet-026c6117b178a9c45,subnet-0839e902f656e8bd1
```

该示例显示一个公有-私有配置，其中，负载均衡器和 EC2 实例在同一公有子网中运行。

Example .ebextensions/vpc.config – 公有-公有

```
option_settings:
   aws:ec2:vpc:
      VPCId: vpc-087a68c03b9c50c84
      AssociatePublicIpAddress: 'true'
      ELBScheme: public
      ELBSubnets: subnet-0fe6b36bcb0ffc462,subnet-032fe3068297ac5b2
      Subnets: subnet-0fe6b36bcb0ffc462,subnet-032fe3068297ac5b2
```

将 Elastic Beanstalk 环境从 EC2-Classic 迁移到 VPC

本主题介绍如何将 Elastic Beanstalk 环境从 EC2-Classic 网络平台迁移到 Amazon Virtual Private Cloud（Amazon VPC）网络的不同选项。

如果您在 2013 年 12 月 4 日之前创建了 AWS 账户，则您的环境可能在某些 AWS 区域 区域中使用 EC2-Classic 网络配置。2013 年 12 月 4 日或之后创建的所有 AWS 账户在每个 AWS 区域都已仅限于 VPC。如果因支持请求而启用了 Amazon EC2-Classic，则会拥有唯一的豁免。

> **Note**
>
> 您可以在 Elastic Beanstalk 控制台的 Configuration overview（配置概览）页面的 Network configuration（网络配置）类别中查看环境的网络配置设置。

为什么应该迁移

Amazon EC2-Classic 将于 2022 年 8 月 15 日之后结束标准支持。为避免工作负载中断，我们建议您在 2022 年 8 月 15 日之前从 Amazon EC2-Classic 迁移到 VPC。我们还要求您以后不要启动任何 Amazon EC2-Classic 上的 AWS 资源，而是使用 Amazon VPC 代替。

当您将 Elastic Beanstalk 环境从 Amazon EC2-Classic 迁移到 Amazon VPC 时，必须创建一个新的 AWS 账户。您还必须在您的新 AWS 账户中重新创建 AWS EC2-Classic 环境。无需为您的环境执行任

何其他配置工作即可使用默认 VPC。如果默认 VPC 不符合您的要求，您可以手动创建自定义 VPC 并将其与您的环境关联。

或者，如果您现有的 AWS 账户拥有无法迁移到新 AWS 账户的资源，则可将 VPC 添加到您的当前账户中。然后，配置您的环境以使用 VPC。

有关更多信息，请参阅 EC2-Classic Networking is Retiring - Here's How to Prepare 博客文章。

将环境从 EC2-Classic 迁移到新的 AWS 账户（推荐）

如果您还没有在 2013 年 12 月 4 日或之后创建的 AWS 账户，则请创建一个新账户。您要将您的环境迁移到此新账户中。

1. 您的新 AWS 账户为其环境提供默认 VPC。如果您不需要创建自定义 VPC，请跳至步骤 2。

   您可以通过以下方式之一创建自定义 VPC：

   - 使用 Amazon VPC 控制台向导，您可以使用其中一个可用的配置选项快速创建 VPC。有关更多信息，请参阅 Amazon VPC 控制台向导配置。
   - 如果您对 VPC 有更具体的要求，请在 Amazon VPC 控制台上创建自定义 VPC。例如，如果您的使用案例需要特定数量的子网，我们建议您这样做。有关更多信息，请参阅 VPC 和子网。
   - 如果您希望在 Elastic Beanstalk 环境中使用 AWS CloudFormation 模板，请使用 GitHub 网站上的 elastic-beanstalk-samples 存储库来创建 VPC。此存储库包括 AWS CloudFormation 模板。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon VPC 结合使用。

   > **Note**
   >
   > 您还可以在使用创建新环境向导在新的 AWS 账户中重新创建环境的同时创建自定义 VPC。如果您使用向导并选择创建自定义 VPC，则该向导将您重新导向到 Amazon VPC 控制台。

2. 在您的新 AWS 账户中创建新环境。我们建议该环境包含与您从中进行迁移的 AWS 账户中的现有环境相同的配置。您可以使用以下方法之一实现这一点。

   > **Note**
   >
   > 如果您的新环境在迁移后必须使用相同的别名记录，则必须首先终止 EC2-Classic 平台上的原始环境。这会释放别名记录以供使用。然而，这样做将导致该环境发生停机，并导致

其他客户可能在终止 EC2-Classic 环境到创建新环境之间选择您的别名记录的风险。有关更多信息，请参阅 终止 Elastic Beanstalk 环境。
对于具有自己的专有域名的环境，CNAME 没有此问题。您只需更新域名系统 (DNS)，即可将请求转发到您的新 CNAME。

- 使用 Elastic Beanstalk 控制台上的创建新环境向导 。该向导提供了一个用于创建自定义 VPC 的选项。如果您不选择创建自定义 VPC，则会分配默认 VPC。
- 使用 Elastic Beanstalk Command Line Interface（EB CLI）在新的 AWS 账户中重新创建您的环境。eb create 命令描述中的其中一个示例演示了如何在自定义 VPC 中创建环境。如果您不提供 VPC ID，环境使用默认 VPC。

  通过使用这种方法，您可以在两个 AWS 账户之间使用保存的配置文件。因此，无需手动输入所有配置信息。然而，您必须使用 eb config save 命令保存要迁移的 EC2-Classic 环境的配置设置。将保存的配置文件复制到新账户环境的新目录中。

  > **Note**
  > 您必须编辑已保存的配置文件中的某些数据，然后才能在新账户中使用它。您必须使用新账户的正确数据更新与先前账户相关的信息。例如，您必须将 AWS Identity and Access Management（IAM）角色的 Amazon Resource Name（ARN）替换为新账户的 IAM 角色 ARN。

  如果您使用带有 `cfg` 的 eb create 命令，系统将使用指定的已保存的配置文件创建新环境。有关更多信息，请参阅 使用 Elastic Beanstalk 保存的配置。

从同一个 AWS 账户中的 EC2-Classic 迁移环境

现有的 AWS 账户可能具有无法迁移到新 AWS 账户的资源。在这种情况下，您必须重新创建环境并为您创建的每个环境手动配置 VPC。

将您的环境迁移到自定义 VPC

先决条件

开始之前，您必须拥有 VPC。您可以通过以下方式之一创建非默认（自定义）VPC：

- 使用 Amazon VPC 控制台向导，您可以使用其中一个可用的配置选项快速创建 VPC。有关更多信息，请参阅 Amazon VPC 控制台向导配置。
- 如果您对 VPC 有更具体的要求，请在 Amazon VPC 控制台上创建自定义 VPC。例如，如果您的使用案例需要特定数量的子网，我们建议您这样做。有关更多信息，请参阅 VPC 和子网。
- 如果您希望在 Elastic Beanstalk 环境中使用 AWS CloudFormation 模板，请使用 GitHub 网站上的 elastic-beanstalk-samples 存储库来创建 VPC。此存储库包括 AWS CloudFormation 模板。有关更多信息，请参阅 将 Elastic Beanstalk 和 Amazon VPC 结合使用。

在以下步骤中，您可以在新环境中配置 VPC 时使用生成的 VPC ID 和子网 ID。

1. 创建与现有环境包含相同配置的新环境。您可以使用以下方法之一实现这一点。

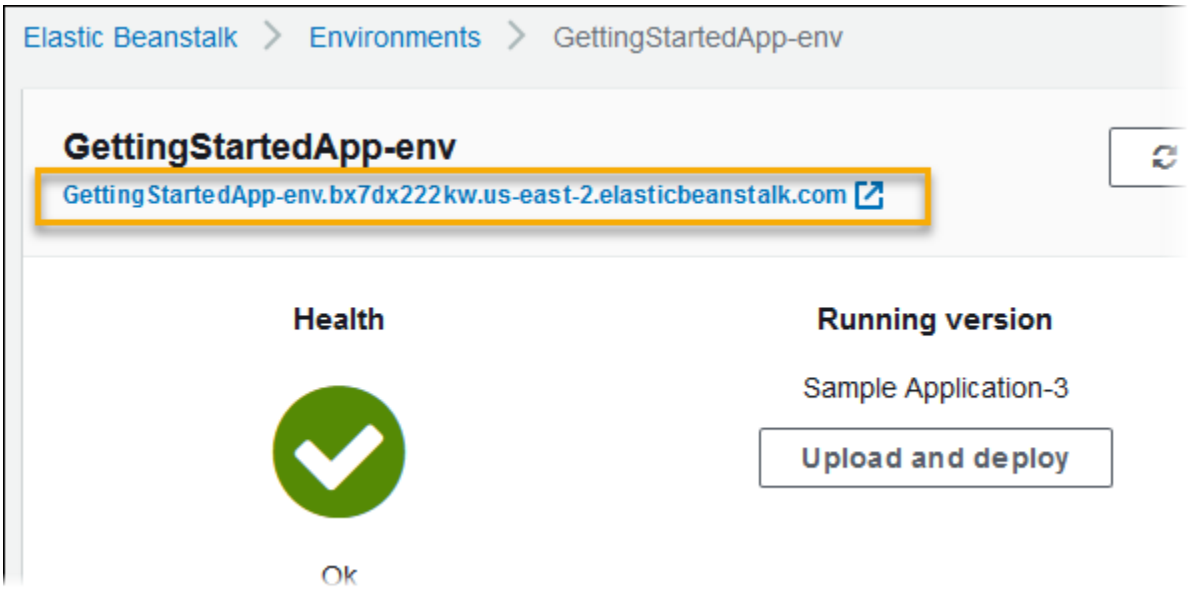
   > **Note**
   >
   > “保存的配置”功能可以帮助您在新账户中重新创建环境。此功能可以保存环境配置，因此您可以在创建或更新其他环境时应用它。有关更多信息，请参阅 使用 Elastic Beanstalk 保存的配置。

   - 使用 Elastic Beanstalk 控制台，在您配置新环境时应用从 EC2-Classic 环境中保存的配置。此配置将使用 VPC。有关更多信息，请参阅 使用 Elastic Beanstalk 保存的配置。
   - 使用 Elastic Beanstalk 命令行界面 (EB CLI)，运行 eb create 命令以重新创建环境。提供原始环境的参数和 VPC 标识符。eb create 命令描述中的其中一个示例演示了如何在自定义 VPC 中创建环境。
   - 使用 AWS Command Line Interface（AWS CLI），并使用 elasticbeanstalk create-environment 命令重新创建环境。使用 VPC 标识符提供原始环境的参数。有关说明，请参阅使用 AWS CLI 创建 Elastic Beanstalk 环境。
2. 将现有环境的别名记录与新环境交换。这样就可以使用熟悉的地址引用创建的新环境。您可以使用 EB CLI 或 AWS CLI。
   - 使用 EB CLI，通过运行 eb swap 命令交换环境别名记录。有关更多信息，请参阅 使用 Elastic Beanstalk 命令行界面 (EB CLI)。
   - 使用 AWS CLI，通过 elasticbeanstalk swap-environment-cnames 命令交换环境别名记录。有关更多信息，请参阅 AWS CLI 命令参考。

您的 Elastic Beanstalk 环境的域名

默认情况下，您的环境可供位于 `elasticbeanstalk.com` 子域的用户使用。在创建环境时，您可以为应用程序选择一个主机名。子域和域自动填充到 *`region`*`.elasticbeanstalk.com`。

为了将用户路由到您的环境，Elastic Beanstalk 将注册一个 CNAME 记录，以指向您环境的负载均衡器。可以在 Elastic Beanstalk 控制台的环境概述页面中查看环境的应用程序 URL 以及 CNAME 的当前值。



在概述页面上选择 URL，或选择导航窗格上的 Go to environment（转至环境）以导航到应用程序的网页。

您可通过将您的环境上的 CNAME 与其他环境的 CNAME 进行交换来更改它。有关说明，请参阅使用 Elastic Beanstalk 进行蓝/绿部署。

如果您拥有一个域名，则可使用 Amazon Route 53 将它解析到您的环境。您可向 Amazon Route 53 购买域名，也可使用从其他提供商处购买的域名。

要向 Route 53 购买域名，请参阅 Amazon Route 53 开发人员指南中的注册新域。

要了解有关使用自定义域的更多信息，请参阅 Amazon Route 53 开发人员指南中的将流量路由到 AWS Elastic Beanstalk 环境。

Important

如果您终止环境，则还必须删除您创建的任何 CNAME 映射，因为其他客户可能会重用可用的主机名。请务必删除指向已终止环境的 DNS 记录，以防出现 悬空 DNS 条目。悬空 DNS 条目可能会使指向您的域的互联网流量出现安全漏洞，此外还可能带来其他风险。

有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的 防止 Route 53 中悬挂委派记录。您还可以访问《AWS 安全博客》中的 针对 Amazon CloudFront 请求的增强域保护，了解有关悬空 DNS 条目的更多信息。

配置 Elastic Beanstalk 环境（高级）

当您创建 AWS Elastic Beanstalk 环境时，Elastic Beanstalk 会预置和配置运行和支持您的应用程序所需的全部AWS资源。除配置您的环境的元数据和更新行为外，您还可以通过为配置选项提供值对这些资源进行自定义。例如，您可能会希望添加 Amazon SQS 队列和有关队列深度的警报，或者添加 Amazon ElastiCache 群集。

大多数配置选项都具有默认值，Elastic Beanstalk 会自动应用这些默认值。您可以使用配置文件、保存的配置、命令行选项或通过直接调用 Elastic Beanstalk API 覆盖这些默认值。EB CLI 和 Elastic Beanstalk 控制台还对某些选项应用建议的值。

在部署应用程序版本的同时，您就可以轻松地自定义环境，只需使用源包将配置文件添加进来即可。当自定义实例上的软件时，使用配置文件而不是创建自定义 AMI 更有帮助，因为不需要维护一系列的 AMI。

部署应用程序时，建议您自定义和配置应用程序所依赖的软件。这些文件可能是应用程序所要求的依赖项，例如 yum 存储库中的其他包，也可能是配置文件，如 httpd.conf 的替代项，用于覆盖 AWS Elastic Beanstalk 默认的特定设置。

主题

配置选项

Elastic Beanstalk 定义了大量配置选项，您可以使用这些选项来配置环境的行为及其包含的资源。配置选项组织到命名空间中，如 `aws:autoscaling:asg`，它定义用于环境的 Auto Scaling 组的选项。

在您创建环境时，Elastic Beanstalk 控制台和 EB CLI 设置一些配置选项，包括您显式设置的选项以及客户端定义的建议值。此外，您还可以在已保存的配置和配置文件中设置配置选项。如果在多个位置设置了同一个选项，则使用的值由优先顺序确定。

配置选项设置可以采用文本格式在环境创建之前进行创建并保存，在创建环境期间使用任何支持的客户端进行应用，以及在环境创建之后进行添加、修改或删除。有关可在这三个阶段的每个阶段中使用配置选项的所有方法的详细划分，请阅读以下主题：

- 在创建环境之前设置配置选项
- 在环境创建期间设置配置选项
- 在环境创建后设置配置选项

有关命名空间和选项的完整列表，包括各自的默认值和支持的值，请参阅面向所有环境的常规选项和特定于平台的选项。

优先顺序

在创建环境期间，配置选项按照以下优先顺序（从最高到最低）从多个来源进行应用：

- 直接应用于环境的设置 – 任何客户端在对 Elastic Beanstalk API 执行创建环境或更新环境操作期间指定的设置，包括 Elastic Beanstalk 控制台、EB CLI、AWS CLI 和开发工具包。Elastic Beanstalk 控制台和 EB CLI 还会为在该级别应用的一些选项应用建议值（除非覆盖）。
- 保存的配置 - 不直接应用于环境的任何选项的设置都从保存的配置加载（如果指定）。
- 配置文件（.ebextensions）– 不直接应用于环境以及未在保存的配置中指定的任何选项的设置都从应用程序源包根文件夹处的 `.ebextensions` 文件夹中的配置文件加载。

  配置文件会按字母顺序执行。例如，`.ebextensions/01run.config` 在 `.ebextensions/02do.config` 之前执行。
- 默认值 - 如果配置选项具有默认值，则它仅当未在以上任何级别上设置选项时才应用。

如果在多个位置定义相同配置选项，则应用优先顺序最高的设置。通过保存的配置或直接应用于环境的设置来应用设置时，设置会作为环境配置的一部分进行存储。可以使用 AWS CLI 或使用 EB CLI 删除这些设置。

配置文件中的设置不直接应用于环境，无法在不修改配置文件并部署新应用程序版本的情况下删除。如果删除使用其他方法之一来应用的设置，则会从源包中的配置文件加载该设置。

例如，假设在创建环境期间，您使用 Elastic Beanstalk 控制台、命令行选项或保存的配置将环境中的最小实例数设置为 5。而应用程序的源包还包含一个将最小实例数设置为 2 的配置文件。

当您创建环境时，Elastic Beanstalk 在 MinSize 命名空间中将 aws:autoscaling:asg 选项设置为 5。如果您随后从环境配置中删除该选项，则会加载配置文件的值，最小实例数设置为 2。如果您随后从源包中删除配置文件并重新部署，则 Elastic Beanstalk 使用默认设置 1。

建议值

Elastic Beanstalk 命令行界面（EB CLI）和 Elastic Beanstalk 控制台为某些配置选项提供了建议值。这些值可能与默认值不同，会在创建环境时在 API 级别上进行设置。建议值使 Elastic Beanstalk 可以改进默认环境配置，而无需对 API 进行无法向后兼容的更改。

例如，EB CLI 和 Elastic Beanstalk 控制台为 EC2 实例类型设置配置选项（InstanceType 命名空间中的 aws:autoscaling:launchconfiguration）。每个客户端都提供不同的方式来覆盖默认设置。在控制台中，您可以从 Create New Environment（创建新环境）向导的 Configuration Details（配置详细信息）页面上的下拉菜单中选择不同的实例类型。使用 EB CLI 时，您可以对 --instance_type 使用 eb create 参数。

由于建议值在 API 级别上进行设置，因此它们会覆盖您在配置文件或保存的配置设置的相同选项值。以下选项会进行设置：

Elastic Beanstalk 控制台

- 命名空间: aws:autoscaling:launchconfiguration

  选项名称：IamInstanceProfile、EC2KeyName、InstanceType
- 命名空间: aws:autoscaling:updatepolicy:rollingupdate

  选项名称：RollingUpdateType 和 RollingUpdateEnabled
- 命名空间: aws:elasticbeanstalk:application

  选项名称：Application Healthcheck URL
- 命名空间: aws:elasticbeanstalk:command

  选项名称：DeploymentPolicy、BatchSize 和 BatchSizeType
- 命名空间: aws:elasticbeanstalk:environment

  选项名称：ServiceRole
- 命名空间: aws:elasticbeanstalk:healthreporting:system

  选项名称：SystemType 和 HealthCheckSuccessThreshold

- 命名空间: aws:elasticbeanstalk:sns:topics

  选项名称：Notification Endpoint
- 命名空间: aws:elasticbeanstalk:sqsd

  选项名称：HttpConnections
- 命名空间: aws:elb:loadbalancer

  选项名称：CrossZone
- 命名空间: aws:elb:policies

  选项名称：ConnectionDrainingTimeout 和 ConnectionDrainingEnabled

EB CLI

- 命名空间: aws:autoscaling:launchconfiguration

  选项名称：IamInstanceProfile、InstanceType
- 命名空间: aws:autoscaling:updatepolicy:rollingupdate

  选项名称：RollingUpdateType 和 RollingUpdateEnabled
- 命名空间: aws:elasticbeanstalk:command

  选项名称：BatchSize 和 BatchSizeType
- 命名空间: aws:elasticbeanstalk:environment

  选项名称：ServiceRole
- 命名空间: aws:elasticbeanstalk:healthreporting:system

  选项名称：SystemType
- 命名空间: aws:elb:loadbalancer

  选项名称：CrossZone
- 命名空间: aws:elb:policies

  选项名称：ConnectionDrainingEnabled

在创建环境之前设置配置选项

AWS Elastic Beanstalk 支持大量配置选项，这些选项使您可以修改应用于环境中的资源的设置。其中一些选项具有默认值，可以覆盖这些值以自定义环境。其他选项可以进行配置以启动附加功能。

Elastic Beanstalk 支持两种用于保存配置选项设置的方法。YAML 或 JSON 格式的配置文件可以放在应用程序的源代码中（位于一个名为 `.ebextensions` 的目录中），并作为应用程序源包的一部分进行部署。您可在本地创建和管理配置文件。

保存的配置是您通过正在运行的环境或 JSON 选项文件创建并存储在 Elastic Beanstalk 中的模板。保存的现有配置也可以进行扩展以创建新配置。

> **Note**
>
> 在配置文件和保存的配置中定义的设置的优先级低于在创建环境期间或在创建环境后配置的设置，包括由 Elastic Beanstalk 控制台和 EB CLI 应用的建议值。有关更多信息，请参阅 优先顺序。

选项还可以在 JSON 文档中指定，并在使用 EB CLI 或 创建或更新环境时直接提供给 Elastic BeanstalkAWS CLI 通过此方法直接提供给 Elastic Beanstalk 的选项会覆盖所有其他方法。

有关可用选项的完整列表，请参阅配置选项。

方法

- 配置文件（.ebextensions）
- 保存的配置
- JSON 文档
- EB CLI 配置

配置文件（`.ebextensions`）

使用 `.ebextensions` 可配置使应用程序正常运行所需的选项，并为可以在更高优先顺序级别上进行覆盖的其他选项提供默认值。`.ebextensions` 中指定的选项的优先顺序最低，可由任何其他级别的设置进行覆盖。

要使用配置文件，请在项目源代码顶层创建一个名为 `.ebextensions` 的文件夹。添加一个扩展名为 `.config` 的文件并按以下方法指定选项：

```
option_settings:
  - namespace:  namespace
    option_name:  option name
    value:  option value
  - namespace:  namespace
    option_name:  option name
    value:  option value
```

例如，以下配置文件将应用程序健康检查 url 设置为 `/health`：

`healthcheckurl.config`

```
option_settings:
  - namespace:  aws:elasticbeanstalk:application
    option_name:  Application Healthcheck URL
    value:  /health
```

在 JSON 中：

```
{
 "option_settings" :
    [
      {
        "namespace" : "aws:elasticbeanstalk:application",
        "option_name" : "Application Healthcheck URL",
        "value" : "/health"
      }
    ]
}
```

这会在 Elastic Beanstalk 环境中配置 Elastic Load Balancing 负载均衡器以向每个 EC2 实例的 `/health` 路径进行 HTTP 请求，从而确定它是否运行状况良好。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

在您的应用程序源包中包含 `.ebextensions` 目录，并将它部署到新的或现有的 Elastic Beanstalk 环境中。

除了 option_settings 之外，配置文件还支持几个部分，用于对在环境中的服务器上运行的软件和文件进行自定义。有关更多信息，请参阅.Ebextensions。

保存的配置

使用 Elastic Beanstalk 控制台、EB CLI 或 创建保存的配置，以保存在创建环境期间或在创建环境后应用于现有环境的设置AWS CLI 保存的配置属于某个应用程序，可以应用于该应用程序的新环境或现有环境。

客户

- Elastic Beanstalk 控制台
- EB CLI
- AWS CLI

Elastic Beanstalk 控制台

创建保存的配置（Elastic Beanstalk 控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Save configuration（保存配置）。
4. 使用屏幕上的对话框完成该操作。

保存的配置存储在 Elastic Beanstalk S3 存储桶中按照应用程序命名的文件夹中。例如，对于账号 123456789012，us-west-2 区域中名为 my-app 的应用程序的配置位于 s3://elasticbeanstalk-us-west-2-123456789012/resources/templates/my-app。

EB CLI

EB CLI 也在 eb config 下提供了子命令，用于与保存的配置进行交互：

创建保存的配置（EB CLI）

1. 保存所附加环境的当前配置：

```
~/project$ eb config save --cfg my-app-v1
```

EB CLI 将配置保存到 ~/project/.elasticbeanstalk/saved_configs/*my-app-v1*.cfg.yml

2. 如果需要，可在本地修改保存的配置。
3. 将保存的配置上传到 S3：

```
~/project$ eb config put my-app-v1
```

AWS CLI

使用aws elasticbeanstalk create-configuration-template从正在运行的环境创建保存的配置

创建保存的配置（AWS CLI）

1. 使用 describe-environments 确定 Elastic Beanstalk 环境的环境 ID：

```
$ aws elasticbeanstalk describe-environments --environment-name my-env
{
    "Environments": [
        {
            "ApplicationName": "my-env",
            "EnvironmentName": "my-env",
            "VersionLabel": "89df",
            "Status": "Ready",
            "Description": "Environment created from the EB CLI using \"eb create
\"",
            "EnvironmentId": "e-vcghmm2zwk",
            "EndpointURL": "awseb-e-v-AWSEBLoa-1JUM8159RA11M-43V6ZI1194.us-
west-2.elb.amazonaws.com",
            "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.2 running Multi-
container Docker 1.7.1 (Generic)",
            "CNAME": "my-env-nfptuqaper.elasticbeanstalk.com",
            "Health": "Green",
            "AbortableOperationInProgress": false,
            "Tier": {
                "Version": " ",
                "Type": "Standard",
```

```
                "Name": "WebServer"
            },
            "HealthStatus": "Ok",
            "DateUpdated": "2015-10-01T00:24:04.045Z",
            "DateCreated": "2015-09-30T23:27:55.768Z"
        }
    ]
}
```

2. 使用 `create-configuration-template` 保存环境的当前配置：

```
$ aws elasticbeanstalk create-configuration-template --environment-id e-vcghmm2zwk
 --application-name my-app --template-name v1
```

Elastic Beanstalk 会将配置保存到 Amazon S3 中的 Elastic Beanstalk 存储桶中。

JSON 文档

如果您使用 AWS CLI 创建和更新环境，则还可以采用 JSON 格式提供配置选项。如果您使用 AWS CLI 创建和管理环境，则 JSON 中的配置文件库会十分有用。

例如，以下 JSON 文档将应用程序健康检查 url 设置为 `/health`：

~/ebconfigs/healthcheckurl.json

```
[
  {
    "Namespace": "aws:elasticbeanstalk:application",
    "OptionName": "Application Healthcheck URL",
    "Value": "/health"
  }
]
```

EB CLI 配置

除了使用 eb config 命令支持保存的配置以及直接环境配置之外，EB CLI 还有一个配置文件，其中包含一个名为 `default_ec2_keyname` 的选项，可以用于指定 Amazon EC2 密钥对以便对环境中的实例进行 SSH 访问。EB CLI 使用此选项在 `EC2KeyName` 命名空间中设置 `aws:autoscaling:launchconfiguration` 配置选项。

~/workspace/my-app/.elasticbeanstalk/config.yml

```
branch-defaults:
  master:
    environment: my-env
  develop:
    environment: my-env-dev
deploy:
  artifact: ROOT.war
global:
  application_name: my-app
  default_ec2_keyname: my-keypair
  default_platform: Tomcat 8 Java 8
  default_region: us-west-2
  profile: null
  sc: git
```

在环境创建期间设置配置选项

使用 Elastic Beanstalk 控制台、EB CLI、AWS CLI、开发工具包或 Elastic Beanstalk API 创建 AWS Elastic Beanstalk 环境时，您可以为配置选项提供值以便自定义环境以及在其中启动的 AWS 资源。

对于一次性配置更改之外的任何更改，可以在本地、源包或 Amazon S3 中存储配置文件。

本主题介绍用于在创建环境期间设置配置选项的所有方法的过程。

客户

- 在 Elastic Beanstalk 控制台中
- 使用 EB CLI
- 使用 AWS CLI

在 Elastic Beanstalk 控制台中

在 Elastic Beanstalk 控制台中创建 Elastic Beanstalk 环境时，您可以使用配置文件、保存的配置以及 Create New Environment（创建新环境）向导中的表单提供配置选项。

方法

- 使用配置文件（.ebextensions）
- 使用保存的配置
- 使用新建环境向导

使用配置文件（**.ebextensions**）

将 .config 文件放在一个名为 .ebextensions 的文件夹内的应用程序源包中。

有关配置文件的详细信息，请参阅.Ebextensions。

```
~/workspace/my-app-v1.zip
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- index.php
`-- styles.css
```

通常在创建环境期间将源包上传到 Elastic Beanstalk。

Elastic Beanstalk 控制台为某些配置选项应用建议值，并为其他选项提供表单字段。通过 Elastic Beanstalk 控制台配置的选项会直接应用于环境，覆盖配置文件中的设置。

使用保存的配置

使用 Elastic Beanstalk 控制台创建新环境时，首先执行的步骤是选择配置。配置可以是预定义配置（通常是某个平台（如 PHP 或 Tomcat）的最新版本），也可以是保存的配置。

在创建环境期间应用保存的配置（Elastic Beanstalk 控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

   > **Note**
   >
   > 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Saved configurations（已保存的配置）。
4. 选择要应用的已保存的配置，然后选择 Launch environment（启动环境）。
5. 继续执行向导以创建环境。

保存的配置将因应用程序而异。有关创建保存的配置的详细信息，请参阅保存的配置。

使用新建环境向导

大多数标准配置选项都在创建新环境向导的配置更多选项页面上。如果您创建 Amazon RDS 数据库或为环境配置 VPC，则会为这些资源提供其他配置选项。

在创建环境期间设置配置选项（Elastic Beanstalk 控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序）。
3. 选择或创建应用程序。
4. 选择 Actions（操作），然后选择 Create environment（创建环境）。
5. 继续执行向导，然后选择配置更多选项。
6. 选择任何 configuration presets（配置预设），然后在一个或多个配置类别中选择 Edit（编辑）以更改一组相关的配置选项。
7. 在完成选项的选择后，选择创建环境。

在新建环境向导中设置的任何选项都直接对环境进行设置，并将覆盖所应用的保存配置或配置文件（`.ebextensions`）中的任何选项设置。可以在创建环境之后使用 EB CLI 或 AWS CLI 删除设置，以允许保存的配置或配置文件中的设置发挥作用。

有关新建环境向导的详细信息，请参阅创建新环境向导。

使用 EB CLI

方法

- 使用配置文件 (.ebextensions)
- 使用保存的配置
- 使用命令行选项

使用配置文件 (**`.ebextensions`**)

将 `.config` 文件放在项目文件夹的 `.ebextensions` 下，以便与应用程序代码一起部署它们。

有关配置文件的详细信息，请参阅.Ebextensions。

```
~/workspace/my-app/
|-- .ebextensions
|   |-- environmentvariables.config
```

```
|   `-- healthcheckurl.config
|-- .elasticbeanstalk
|   `-- config.yml
|-- index.php
`-- styles.css
```

使用 eb create 创建环境并将源代码部署到其中。

```
~/workspace/my-app$ eb create my-env
```

使用保存的配置

要在使用 eb create 创建环境时应用保存的配置，请使用 --cfg 选项。

```
~/workspace/my-app$ eb create --cfg savedconfig
```

可以将保存的配置存储在项目文件夹中或 Amazon S3 上的 Elastic Beanstalk 存储位置中。在以上示例中，EB CLI 首先在文件夹 savedconfig.cfg.yml 中查找名为 .elasticbeanstalk/saved_configs/ 的保存的配置文件。在使用 .cfg.yml 应用保存的配置时，不要包含文件扩展名（--cfg）。

```
~/workspace/my-app/
|-- .ebextensions
|   `-- healthcheckurl.config
|-- .elasticbeanstalk
|   |-- saved_configs
|   |   `-- savedconfig.cfg.yml
|   `-- config.yml
|-- index.php
`-- styles.css
```

如果 EB CLI 未在本地找到配置，则它会在 Amazon S3 中的 Elastic Beanstalk 存储位置中进行查找。有关创建、编辑和上传保存的配置的详细信息，请参阅保存的配置。

使用命令行选项

EB CLI eb create 命令具有几个选项，可以用于在创建环境期间设置配置选项。可以使用这些选项来将 RDS 数据库添加到环境、配置 VPC 或覆盖建议值。

例如，EB CLI 在默认情况下使用 t2.micro 实例类型。要选择不同的实例类型，请使用 --instance_type 选项。

```
$ eb create my-env --instance_type t2.medium
```

要创建 Amazon RDS 数据库实例并将它挂载到环境，请使用 `--database` 选项。

```
$ eb create --database.engine postgres --database.username dbuser
```

如果您省略环境名称、数据库密码或创建环境所需的任何其他参数，则 EB CLI 会提示您输入它们。

有关可用选项的完整列表以及用法示例，请参阅 eb create。

使用 AWS CLI

当您通过 AWS CLI 使用 `create-environment` 命令创建 Elastic Beanstalk 环境时，AWS CLI 不应用任何建议值。指定源包中的配置文件中定义了所有配置选项。

方法

- 使用配置文件 (.ebextensions)
- 使用保存的配置
- 使用命令行选项

使用配置文件 (`.ebextensions`)

要使用 AWS CLI 将配置文件应用于您创建的环境，请在上传到 Amazon S3 的应用程序源包中包含它们。

有关配置文件的详细信息，请参阅.Ebextensions。

```
~/workspace/my-app-v1.zip
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- index.php
`-- styles.css
```

使用 上传应用程序源包并创建环境AWS CLI

1. 如果您在 Amazon S3 中还没有 Elastic Beanstalk 存储桶，请使用 `create-storage-location` 创建一个。

```
$ aws elasticbeanstalk create-storage-location
{
    "S3Bucket": "elasticbeanstalk-us-west-2-123456789012"
}
```

2. 将应用程序源包上传到 Amazon S3。

```
$ aws s3 cp sourcebundle.zip s3://elasticbeanstalk-us-west-2-123456789012/my-app/
sourcebundle.zip
```

3. 创建应用程序版本。

```
$ aws elasticbeanstalk create-application-version --application-name my-app --
version-label v1 --description MyAppv1 --source-bundle S3Bucket="elasticbeanstalk-
us-west-2-123456789012",S3Key="my-app/sourcebundle.zip" --auto-create-application
```

4. 创建环境。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-
name my-env --version-label v1 --solution-stack-name "64bit Amazon Linux 2015.03
 v2.0.0 running Tomcat 8 Java 8"
```

使用保存的配置

要在创建过程中将保存的配置应用于环境，请使用 `--template-name` 参数。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-name
 my-env --template-name savedconfig --version-label v1
```

指定保存的配置时，不要同时指定解决方案堆栈名称。保存的配置已指定解决方案堆栈，如果您尝试同时使用两个选项，则 Elastic Beanstalk 将返回错误。

使用命令行选项

使用 `--option-settings` 参数可采用 JSON 格式指定配置选项。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-name
 my-env --version-label v1 --template-name savedconfig --option-settings '[
  {
```

```
    "Namespace": "aws:elasticbeanstalk:application",
    "OptionName": "Application Healthcheck URL",
    "Value": "/health"
  }
]
```

要从文件加载 JSON，请使用 `file://` 前缀。

```
$ aws elasticbeanstalk create-environment --application-name my-app --environment-
name my-env --version-label v1 --template-name savedconfig --option-settings file://
healthcheckurl.json
```

Elastic Beanstalk 会将您使用 `--option-settings` 选项指定的选项设置直接应用于环境。如果在保存的配置或配置文件中指定了相同选项，则 `--option-settings` 会覆盖这些值。

在环境创建后设置配置选项

您可以修改正在运行的环境的选项设置，方法是应用保存的配置，上传带有配置文件（`.ebextensions`）的新的源包或使用 JSON 文档。EB CLI 和 Elastic Beanstalk 控制台还具有客户端特定的功能，可用于设置和更新配置选项。

当您设置或更改某一配置选项时，根据更改的严重性，可能会触发完整环境更新。例如，如果对 `aws:autoscaling:launchconfiguration` 中的选项（如 `InstanceType`）进行更改，则需要重新预置环境中的 Amazon EC2 实例。这将触发滚动更新。其他配置更改无需中断或重新配置即可应用。

您可以使用 EB CLI 或 AWS CLI 命令从环境中删除选项设置。通过删除在 API 级别直接对环境设置的选项，可使配置文件中的设置（否则被直接应用于环境的设置屏蔽）将生效。

通过其他某种配置方法直接对环境设置相同的选项可以覆盖保存的配置和配置文件中的相应设置。但只有应用更新的保存的配置或配置文件才能完全删除这些设置。如果选项在保存的配置、配置文件中未设置并且也没有直接对环境设置，则应用默认值（如果有默认值）。有关更多信息，请参阅 优先顺序。

客户

- Elastic Beanstalk 控制台
- EB CLI
- 这些区域有： AWS CLI

Elastic Beanstalk 控制台

您可以在 Elastic Beanstalk 控制台中更新配置选项设置，方法是部署包含配置文件的应用程序源包，应用保存的配置，或者在环境管理控制台中使用 Configuration（配置）页面直接修改环境。

方法

- 使用配置文件 (.ebextensions)
- 使用保存的配置
- 使用 Elastic Beanstalk 控制台

使用配置文件 (`.ebextensions`)

更新源目录中的配置文件，创建新的源包，并将新版本部署到您的 Elastic Beanstalk 环境以应用更改。

有关配置文件的详细信息，请参阅.Ebextensions。

部署源包

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在环境概述页面上，选择 Upload and deploy（上传和部署）。
4. 使用屏幕上的对话框上传源包。
5. 选择 Deploy（部署）。
6. 部署完成后，选择站点 URL 以在新选项卡中打开您的网站。

对配置文件进行的更改不会覆盖保存的配置中的选项设置或在 API 级别直接应用于环境的设置。有关详细信息，请参阅优先顺序。

使用保存的配置

对正在运行的环境应用保存的配置以便应用它所定义的选项设置。

将保存的配置应用于正在运行的环境（Elastic Beanstalk 控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Saved configurations (已保存的配置)。
4. 选择要应用的已保存的配置，然后选择 Load（加载）。
5. 选择一个环境，然后选择加载。

保存的配置中定义的设置会覆盖配置文件中的设置，但会被使用环境管理控制台配置的设置覆盖。

有关创建保存的配置的详细信息，请参阅保存的配置。

使用 Elastic Beanstalk 控制台

Elastic Beanstalk 控制台在每个环境的 Configuration（配置）页面上提供多个配置选项。

在正在运行的环境中更改配置选项（Elastic Beanstalk 控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 找到要编辑的配置页面：
   - 如果您看到感兴趣的选项，或者您知道它所在的配置类别，请在它的配置类别中选择 Edit（编辑）。
   - 要查找一个选项，请打开表视图，然后在搜索框中输入搜索词。在您键入时，列表将会变短，并且仅显示与搜索词匹配的选项。

     在看到要查找的选项时，请在包含该选项的配置类别中选择 Edit（编辑）。

5. 更改设置，然后选择保存。
6. 根据需要，在其他配置类别中重复前面的两个步骤。
7. 选择 Apply。

在环境管理控制台中对配置选项所做的更改将直接应用于环境。这些更改覆盖配置文件或保存的配置中的相同选项的设置。有关详细信息，请参阅优先顺序。

有关使用 Elastic Beanstalk 控制台更改运行的环境上的配置选项的详细信息，请参阅配置 Elastic Beanstalk 环境下面的主题。

EB CLI

您可以使用 EB CLI 更新配置选项设置，方法是部署包含配置文件的源代码，应用来自保存的配置的设置，或使用 eb config 命令直接修改环境配置。

方法

- 使用配置文件 (.ebextensions)
- 使用保存的配置
- 使用 eb config
- 使用 eb setenv

使用配置文件 (`.ebextensions`)

将 `.config` 文件放在项目文件夹的 `.ebextensions` 下，以便与应用程序代码一起部署它们。

有关配置文件的详细信息，请参阅.Ebextensions。

```
~/workspace/my-app/
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- .elasticbeanstalk
|   `-- config.yml
|-- index.php
`-- styles.css
```

使用 eb deploy 部署您的源代码。

```
~/workspace/my-app$ eb deploy
```

使用保存的配置

您可以使用 eb config 命令对正在运行的环境应用保存的配置。使用 `--cfg` 选项和保存的配置的名称可将其设置应用于环境。

```
$ eb config --cfg v1
```

在本例中，v1 是之前创建并保存的配置文件的名称。

使用此命令应用于环境的设置会覆盖在环境创建期间应用的设置以及在应用程序源包中的配置文件中定义的设置。

使用 eb config

通过 EB CLI 的 eb config 命令，您可以使用文本编辑器直接设置或删除对环境的选项设置。

当您运行 eb config 时，EB CLI 会显示从所有源（包括配置文件、保存的配置、建议值、直接对环境设置的选项和 API 默认值）应用于环境的设置。

> **Note**
>
> eb config 不会显示环境属性。要设置可在您的应用程序中读取的环境属性，请使用 eb setenv。

下面的示例显示在 `aws:autoscaling:launchconfiguration` 命名空间中应用的设置。这些设置包括：

- EB CLI 在环境创建期间对 IamInstanceProfile 和 InstanceType 应用的两个建议值。
- 在创建期间根据存储库配置直接对环境设置的选项 EC2KeyName。
- 其他选项的 API 默认值。

```
ApplicationName: tomcat
DateUpdated: 2015-09-30 22:51:07+00:00
EnvironmentName: tomcat
SolutionStackName: 64bit Amazon Linux 2015.03 v2.0.1 running Tomcat 8 Java 8
settings:
...
aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: null
    EC2KeyName: my-key
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    ImageId: ami-1f316660
    InstanceType: t2.micro
...
```

使用 eb config 设置或更改配置选项

1. 运行 eb config 查看您的环境配置。

   ```
   ~/workspace/my-app/$ eb config
   ```

2. 使用默认文本编辑器更改任意设置值.

   ```
   aws:autoscaling:launchconfiguration:
       BlockDeviceMappings: null
       EC2KeyName: my-key
       IamInstanceProfile: aws-elasticbeanstalk-ec2-role
       ImageId: ami-1f316660
       InstanceType: t2.medium
   ```

3. 保存临时配置文件并退出。
4. EB CLI 更新您的环境配置。

使用 eb config 设置配置选项会覆盖所有其他来源的设置。

您也可以使用 eb config 删除您的环境中的选项。

删除配置选项（EB CLI）

1. 运行 eb config 查看您的环境配置。

```
~/workspace/my-app/$ eb config
```

2. 将所有显示的值都替换为字符串 null。您也可以删除包含要删除选项的整行。

```
aws:autoscaling:launchconfiguration:
    BlockDeviceMappings: null
    EC2KeyName: my-key
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    ImageId: ami-1f316660
    InstanceType: null
```

3. 保存临时配置文件并退出。
4. EB CLI 更新您的环境配置。

通过 eb config 从环境中删除选项，可以使应用程序源包中配置文件中的相应选项设置生效。有关详细信息，请参阅优先顺序。

使用 eb setenv

要使用 EB CLI 设置环境属性，请使用 eb setenv。

```
~/workspace/my-app/$ eb setenv ENVVAR=TEST
INFO: Environment update is starting.
INFO: Updating environment my-env's configuration settings.
INFO: Environment health has transitioned from Ok to Info. Command is executing on all
 instances.
INFO: Successfully deployed new configuration to environment.
```

此命令在 aws:elasticbeanstalk:application:environment 命名空间中设置环境属性。通过 eb setenv 设置的环境属性经过短暂的更新过程即可用于您的应用程序。

使用 eb printenv 可查看您的环境中设置的环境属性。

```
~/workspace/my-app/$ eb printenv
 Environment Variables:
     ENVVAR = TEST
```

这些区域有： AWS CLI

您可以使用 AWS CLI 更新配置选项设置，方法是部署包含配置文件的源包，应用远程存储的保存的配置，或使用 `aws elasticbeanstalk update-environment` 命令直接修改环境。

方法

- 使用配置文件 (.ebextensions)
- 使用保存的配置
- 使用命令行选项

使用配置文件 (**.ebextensions**)

要通过 AWS CLI 对正在运行的环境应用配置文件，请将它们包含在要上传到 Amazon S3 的应用程序源包中。

有关配置文件的详细信息，请参阅.Ebextensions。

```
~/workspace/my-app-v1.zip
|-- .ebextensions
|   |-- environmentvariables.config
|   `-- healthcheckurl.config
|-- index.php
`-- styles.css
```

上传应用程序源包并将其应用于正在运行的环境（AWS CLI）

1. 如果您在 Amazon S3 中还没有 Elastic Beanstalk 存储桶，请使用 `create-storage-location` 创建一个：

   ```
   $ aws elasticbeanstalk create-storage-location
   {
       "S3Bucket": "elasticbeanstalk-us-west-2-123456789012"
   }
   ```

2. 将应用程序源包上传到 Amazon S3。

   ```
   $ aws s3 cp sourcebundlev2.zip s3://elasticbeanstalk-us-west-2-123456789012/my-app/
   sourcebundlev2.zip
   ```

3. 创建应用程序版本。

```
$ aws elasticbeanstalk create-application-version --application-
name my-app --version-label v2 --description MyAppv2 --source-bundle
 S3Bucket="elasticbeanstalk-us-west-2-123456789012",S3Key="my-app/
sourcebundlev2.zip"
```

4. 更新环境。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --version-label
 v2
```

使用保存的配置

通过在 `--template-name` 命令中使用 `aws elasticbeanstalk update-environment` 选项可以对正在运行的环境应用保存的配置。

保存的配置必须位于您的 Elastic Beanstalk 存储桶中，位于 `resources/templates` 下以您的应用程序命名的路径中。例如，对于账户 123456789012，美国西部（俄勒冈）区域（us-west-2）中的 v1 应用程序的 my-app 模板位于 `s3://elasticbeanstalk-us-west-2-123456789012/resources/templates/my-app/v1`

对正在运行的环境应用保存的配置（AWS CLI）

- 使用 `update-environment` 选项在 `--template-name` 调用中指定保存的配置。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --template-
name v1
```

Elastic Beanstalk 在您使用 `aws elasticbeanstalk create-configuration-template` 创建时将保存的配置放在此位置。您也可以在本地修改保存的配置并自行将它们放在此处。

使用命令行选项

使用 JSON 文档更改配置选项（AWS CLI）

1. 在本地文件中以 JSON 格式定义您的选项设置。
2. 使用 `update-environment` 选项运行 `--option-settings`。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --option-
settings file://~/ebconfigs/as-zero.json
```

在本例中，as-zero.json 定义使用最少和最大零个实例配置环境的选项。这会停止环境中的实例而不终止环境。

~/ebconfigs/as-zero.json

```
[
    {
        "Namespace": "aws:autoscaling:asg",
        "OptionName": "MinSize",
        "Value": "0"
    },
    {
        "Namespace": "aws:autoscaling:asg",
        "OptionName": "MaxSize",
        "Value": "0"
    },
    {
        "Namespace": "aws:autoscaling:updatepolicy:rollingupdate",
        "OptionName": "RollingUpdateEnabled",
        "Value": "false"
    }
]
```

> **Note**
>
> 使用 update-environment 设置配置选项会覆盖所有其他来源的设置。

您也可以使用 update-environment 删除您的环境中的选项。

删除配置选项（AWS CLI）

- 使用 update-environment 选项运行 --options-to-remove 命令。

```
$ aws elasticbeanstalk update-environment --environment-name my-env --options-to-
remove Namespace=aws:autoscaling:launchconfiguration,OptionName=InstanceType
```

通过 `update-environment` 从环境中删除选项，可以使应用程序源包中配置文件中的相应选项设置生效。如果有一个选项不是使用这些方法中的任何方法配置的，则在有 API 默认值时就会使用 API 默认值。有关详细信息，请参阅优先顺序。

面向所有环境的常规选项

命名空间

- aws:autoscaling:asg
- aws:autoscaling:launchconfiguration
- aws:autoscaling:scheduledaction
- aws:autoscaling:trigger
- aws:autoscaling:updatepolicy:rollingupdate
- aws:ec2:instances
- aws:ec2:vpc
- aws:elasticbeanstalk:application
- aws:elasticbeanstalk:application:environment
- aws:elasticbeanstalk:cloudwatch:logs
- aws:elasticbeanstalk:cloudwatch:logs:health
- aws:elasticbeanstalk:command
- aws:elasticbeanstalk:environment
- aws:elasticbeanstalk:environment:process:default
- aws:elasticbeanstalk:environment:process:process_name
- aws:elasticbeanstalk:environment:proxy:staticfiles
- aws:elasticbeanstalk:healthreporting:system
- aws:elasticbeanstalk:hostmanager
- aws:elasticbeanstalk:managedactions
- aws:elasticbeanstalk:managedactions:platformupdate
- aws:elasticbeanstalk:monitoring
- aws:elasticbeanstalk:sns:topics
- aws:elasticbeanstalk:sqsd
- aws:elasticbeanstalk:trafficsplitting
- aws:elasticbeanstalk:xray

- aws:elb:healthcheck
- aws:elb:loadbalancer
- aws:elb:listener
- aws:elb:listener:listener_port
- aws:elb:policies
- aws:elb:policies:policy_name
- aws:elbv2:listener:default
- aws:elbv2:listener:listener_port
- aws:elbv2:listenerrule:rule_name
- aws:elbv2:loadbalancer
- aws:rds:dbinstance

aws:autoscaling:asg

配置环境的 Auto Scaling 组。有关更多信息，请参阅the section called “Auto Scaling 组”。

命名空间: **`aws:autoscaling:asg`**

名称	描述	默认	有效值

Availability Zones	可用区 (AZ) 是一个 AWS 区域内的不同位置，经过精心设计，可与其他可用区中的故障隔离开来。它们为同一区域中的其他 AZ 提供低成本、低延迟的网络连接。选择实例的可用区数量。	`Any`	`Any`
			`Any 1`
			`Any 2`
			`Any 3`

Cooldown	冷却时间有助于防止 Amazon EC2 Auto Scaling 在先前活动产生明显影响前启动其他扩展活动。冷却时间是一个调整活动完成后、另一个调整活动开始前的时长（秒）。	`360`	`0` 到 `10000`
Custom Availability Zones	定义实例的可用区。	无	`us-east-1a` `us-east-1b`

名称	描述	默认	有效值

			`us-east-1c`
			
			`us-east-1d`
			
			`us-east-1e`
			
			`eu-central-1`

EnableCapacityRebalancing	指定是否为 Auto Scaling 组中的 Spot 实例启用容量再平衡功能。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的容量再平衡。
	
	只有在 `aws:ec2:instances` 命名空间里将 `EnableSpot` 设置为 `true`，并且 Auto Scaling 组中至少有一个 Spot 实例时，此选项才会相关联。
		`false`	`true`
			
			`false`

MinSize	您希望 Auto Scaling 组中拥有的最小实例数。	`1`	1 到 `10000`
MaxSize	您希望 Auto Scaling 组中拥有的最大实例数。	`4`	1 到 `10000`

aws:autoscaling:launchconfiguration

配置环境的 Amazon Elastic Compute Cloud（Amazon EC2）实例。

使用 Amazon EC2 启动模板或使用 Auto Scaling 组启动配置资源来创建环境的实例。以下选项适用于这两种资源类型。

有关更多信息，请参阅 the section called “Amazon EC2 实例”。您还可以在亚马逊 EC2 用户指南的亚马逊 EBS 章节中参考有关亚马逊弹性区块存储 (EBS) 的更多信息。

命名空间: **`aws:autoscaling:launchconfiguration`**

名称	描述	默认	有效值

DisableIMDSv1	设置为 `true` 可禁用实例元数据服务版本 1（IMDSv1）。
	
	根据平台操作系统，您的环境的实例默认如下所示：
	
	• Windows server、AL2 及更早版本 – 同时启用 IMDSv1 和 IMDSv2
	• AL2023 – 仅启用 IMDSv2
	
	有关更多信息，请参阅配置实例元数据服务（Amazon Linux）。或配置实例元数据服务（Windows 服务器）。
		`false` – 基于 Windows 服务器、Amazon Linux 2 及更早版本的平台
		
		`true` – 基于 Amazon Linux 2023 的平台
			`true`
			
			`false`

EC2KeyName	您可以使用密钥对安全地登录 EC2 实例。 **Note** 如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。	无	
IamInstanceProfile	实例配置文件允许 AWS Identity and Access Management (IAM) 用户和 AWS 服务访问临时安全证书以进行 AWS API 调用。指定实例配置文件的名称或其 ARN。 示例：	无	实例配置文件名称或 ARN。

名称	描述	默认	有效值
	• `aws-elasticbeanstalk-ec2-role` • `arn:aws:iam::123456789012:instance-profile/aws-elasticbeanstalk-ec2-role` **Note** 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。		
ImageId	通过指定自定义 AMI ID，您可以覆盖默认的 Amazon Machine Image（AMI）。 例如：`ami-1f316660`	无	

名称	描述	默认	有效值
InstanceType	用于在 Elastic Beanstalk 环境中运行您的应用程序的实例类型。 **Important** `InstanceType` 选项已过时。它被 `InstanceTypes` 命名空间中更新、功能更强大的 `aws:ec2:instances` 选项所取代。您可以使用新选项为环境指定一个或多个实例类型的列表。该列表上的第一个值等于此处描述的 `aws:autoscaling:launchconfiguration` 命名空间中包含的 `InstanceType` 选项的值。我们建议您使用新选项指定实例类型。如果指定，则新选项优先于前一个选项。有关更多信息，请参阅 the section called “aws:ec2:instances 命名空间”。 可用的实例类型取决于使用的可用区和区域。如果您选择子网，则包含该子网的可用区将决定可用的实例类型。 • Elastic Beanstalk 不支持 Amazon EC2 Mac 实例类型。 • 有关 Amazon EC2 实例系列和类型的更多信息，请参阅 Amazon EC2 用户指南中的实例类型或亚马逊 EC2 用户指南中的实例类型。	因账户和区域而异。	一种 EC2 实例类型 因账户、区域和可用区而异。您可以获取通过这些值筛选的 Amazon EC2 实例类型的列表。有关更多信息，请参阅 Amazon EC2 用户指南中的可用实例类型或 Amazon EC2 用户指南中的可用实例类型。

名称	描述	默认	有效值
	• 有关各区域可用实例类型的更多信息，请参阅 Amazon EC2 用户指南中的可用实例类型或 Ama zon EC2 用户指南中的可用实例类型。 **Note** 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。		

名称	描述	默认	有效值

LaunchTemplateTagPropagationEnabled	设置为 `true`，以允许将环境标签传播到为环境预调配的特定资源的启动模板。
	
	Elastic Beanstalk 只能将标签传播到以下资源的启动模板：
	
	• EBS 卷
	• EC2 实例
	• EC2 网络接口
	• AWS CloudFormation 启动定义资源的模板
	
	之所以存在此限制，是因为 CloudFormation 仅允许在为特定资源创建模板时使用标签。有关更多信息，请参阅TagSpecification《AWS CloudFormation 用户指南》。
	
	Important
	
	• 将现有环境的此选项值从更改 `false` 为 `true` 对于先前存在的标签可能是一项重大更改。
	• 启用此功能后，传播标签将需要更换 EC2，这可能会导致停机。您可以启用滚动更新以批量应用配置更改，并防止在更新过程中出现停机。有关更多信息，请参阅 配置更改。
		`false`	`true`
			
			`false`

名称	描述	默认	有效值
	有关启动模板的更多信息，请参阅以下内容： • 《Amazon EC2 Auto Scaling 用户指南》中的启动模板。 • 《AWS CloudFormation 用户指南》中的使用模板 • 《AWS CloudFormation 用户指南》中的 Elastic Beanstalk 模板片段 有关此选项的更多信息，请参阅 标签传播到启动模板。		
MonitoringInterval	您希望返回 Amazon CloudWatch 指标的时间间隔（以分钟为单位）。	`5 minute`	`1 minute` `5 minute`
SecurityGroups	列出分配给 Auto Scaling 组中的 EC2 实例的 Amazon EC2 安全组，以便为这些实例定义防火墙规则。 您可以提供一串以逗号分隔的值，其中包含现有 Amazon EC2 安全组的名称或对模板中创建的 AWS::EC2::SecurityGroup 资源的引用。安全组名称区分大小写。 如果您将 Amazon Virtual Private Cloud（Amazon VPC）与 Elastic Beanstalk 结合使用，以便在 Virtual Private Cloud（VPC）中启动实例，请指定安全组 ID 而不是安全组名称。	`elasticbeanstalk-default`	

名称	描述	默认	有效值
SSHSourceRestriction	用于锁定 SSH 对环境的访问权限。例如，您可以锁定 SSH 对 EC2 实例的访问权限，以便只有堡垒主机才能访问私有子网中的实例。 该字符串采用以下形式： *protocol*, *fromPort*, *toPort*, *source_restriction* *protocol* 入口流量规则的协议。 *fromPort* 起始端口号。 *toPort* 结尾端口号。 *source_restriction* 流量必须路由通过的 CIDR 范围或安全组的名称。要从另一账户（仅限 EC2-Classic，必须位于同一区域中）中指定安全组，请在安全组名称前包含账户 ID。采用以下格式：*other_account_id*/*security_group_name*　如果您将 Amazon Virtual Private Cloud（Amazon VPC）与 Elastic Beanstalk 结合使用，以便在 Virtual Private Cloud（VPC）中启动实例，请指定安全组 ID 而不是安全组名称。	无	

名称	描述	默认	有效值
	示例：`tcp, 22, 22, 54.240.196.185/32` 例如：`tcp, 22, 22, my-security-group` 示例（EC2-Classic）：`tcp, 22, 22, 123456789012/their-security-group` 示例（VPC）：`tcp, 22, 22, sg-903004f8`		

名称	描述	默认	有效值
BlockDeviceMappings	在 Auto Scaling 组中的所有实例上挂载其他 Amazon EBS 卷或实例存储卷。 映射实例存储卷时，您只需将设备名称映射到卷名称。但是，我们建议在映射 Amazon EBS 卷时，另外指定以下部分或全部字段（每个字段必须用冒号分隔）： • 快照 ID • 大小（GB） • 终止时删除（`true` 或 `false`） • 存储类型（仅适用于 `gp3`、`gp2`、`standard`、`st1`、`sc1` 或 `io1`） • IOPS（仅适用于 `gp3` 或者 `io1`） • 吞吐量（仅适用于 `gp3`） 下面的示例附加三个 Amazon EBS 卷：一个空白 100GB gp2 卷和一个快照、一个具有 2000 个预配置 IOPS 的空白 20GB io1 卷以及一个实例存储卷 `ephemeral0` 。如果实例类型支持，则可以附加多个实例存储卷。 `/dev/sdj=:100:true:gp2,/dev/sdh=snap-51eef269,/dev/sdi=:20:true:io1:2000,/dev/sdb=ephemeral0`	无	• 大小 — 必须在 500 到 16384 GiB 之间 • 吞吐量 — 必须在每秒 125 到 1000 兆字节之间（MiB/s）

名称	描述	默认	有效值
RootVolumeType	用于已附加到环境的 EC2 实例的根 Amazon EBS 卷的卷类型（机械硬盘、通用型 SSD 或预置 IOPS SSD）。	因平台而异。	对于机械硬盘存储，为 `standard`。 对于通用型 SSD，为`gp2` 或 `gp3`。 对于预配置的 IOPS SSD，为 `io1`。
RootVolumeSize	根 Amazon EBS 卷的存储容量（以完整 GB 为单位）。 如果将 `RootVolumeType` 设置为预配置的 IOPS SSD，则是必需的。 例如 。`"64"`	对于机械硬盘存储和通用型 SSD，因平台而异。 对于预配置的 IOPS SSD，为“无”。	对于通用型和预配置的 IOPS SSD，为 `10` 至 `16384` GB。 对于机械硬盘存储，为 `8` 至 `1024` GB。
RootVolumeIOPS	预置 IOPS SSD 根卷或通用型 `gp3` SSD 根卷所需的每秒输入/输出操作数（IOPS）。 IOPS 与卷大小的最大比率为 500:1。例如，IOPS 为 3000 的卷至少必须为 6 GiB。	无	对于 io1 预置 IOPS 固态硬盘根卷，范围为 `100` 到 `20000`。 对于通用型 `gp3` 固态硬盘根卷，范围为 `3000` 到 `16000`。

名称	描述	默认	有效值
RootVolumeThroughput	已附加到环境的 EC2 实例的 Amazon EBS 根卷所需的每秒吞吐量兆字节数（MiB/s）。 **Note** 此选项仅适用于 gp3 存储类型。	无	125 到 1000

aws:autoscaling:scheduledaction

为环境的 Auto Scaling 组配置计划操作。对于每个操作，除了指定选项名称、命名空间和每个设置的值之外，还要指定 `resource_name`。有关示例，请参阅 aws:autoscaling:scheduledaction 命名空间。

命名空间: **`aws:autoscaling:scheduledaction`**

名称	描述	默认	有效值
StartTime	对于一次性操作，请选择运行操作的日期和时间。对于重复操作，请选择激活操作的时间。	无	跨所有计划的扩展操作的唯一 ISO-8601 时间戳。
EndTime	您希望计划的扩展操作停止重复的将来日期和时间（采用 UTC/GMT 时区）。如果未指定 EndTime，则操作将根据`Recurrence` 表达式重复。 例如：`2015-04-28T04:07:2Z` 当计划的操作结束时，Amazon EC2 Auto Scaling 不会自动恢复到其以前的设置。配置第二个计划操作，以根据需要返回原始设置。	无	跨所有计划的扩展操作的唯一 ISO-8601 时间戳。

名称	描述	默认	有效值
MaxSize	运行操作时要应用的最大实例计数。	无	`0` 到 `10000`
MinSize	运行操作时要应用的最小实例计数。	无	`0` 到 `10000`
DesiredCapacity	为 Auto Scaling 组设置初始所需容量。在应用计划的操作后，触发器将根据其设置调整所需容量。	无	`0` 到 `10000`
Recurrence	您希望计划操作发生的频率。如果不指定循环，则扩展操作仅发生一次，如 `StartTime` 所指定。	无	Cron 表达式。

Suspend	设置为 `true` 可临时停用重复的计划操作。	`false`	`true`
			
			`false`

aws:autoscaling:trigger

为环境的 Auto Scaling 组配置扩展触发器。

> **Note**
>
> 此命名空间中的三个选项确定在触发器启用之前触发器指标可超出其定义的限制多长时间。这些选项具有相关性，如下所述：
> `BreachDuration = Period * EvaluationPeriods`
> 这些选项的默认值（分别是 5、5 和 1）满足此等式。如果您指定不一致的值，Elastic Beanstalk 可能会修改其中某个值，以便等式仍然成立。

命名空间: **`aws:autoscaling:trigger`**

名称	描述	默认	有效值
BreachDuration	调用触发器前，指标可以超出所定义限制（如 `UpperThreshold` 和	`5`	`1` 到 `600`

名称	描述	默认	有效值
	LowerThreshold 所指定）的时间（单位：分钟）。		
LowerBreachScaleIncrement	执行扩展活动时要移除的 Amazon EC2 实例的数量。	-1	
LowerThreshold	如果测量值低于该违例持续时间值，则会调用触发器。	2000000	0 到 20000000
MeasureName	用于 Auto Scaling 触发器的指标。 **Note** HealthyHostCount 、UnhealthyHostCount 和 TargetResponseTime 仅适用于具有专用负载均衡器的环境。对于配置了共享负载均衡器的环境，这些值不是有效指标值。有关负载均衡器类型的更多信息，请参阅 Elastic Beanstalk 环境的负载均衡器。	NetworkOut	CPUUtilization NetworkIn NetworkOut DiskWriteOps DiskReadBytes DiskReadOps DiskWriteBytes Latency RequestCount HealthyHostCount UnhealthyHostCount TargetResponseTime

名称	描述	默认	有效值
Period	指定 Amazon CloudWatch 衡量触发器指标的频率。此值是两个连续时间段之间的分钟数。	5	1 到 600
EvaluationPeriods	用于确定是否违例的连续评估期的数量。	1	1 到 600
Statistic	触发器使用的统计数据，例如 Average。	Average	Minimum Maximum Sum Average
Unit	触发器度量单位，例如 Bytes。	Bytes	Seconds Percent Bytes Bits Count Bytes/Second Bits/Second Count/Second None
UpperBreachScaleIncrement	执行扩展活动时，指定要添加的 Amazon EC2 实例的数量。	1	
UpperThreshold	如果测量值高于该违例持续时间值，则会调用触发器。	6000000	0 到 20000000

aws:autoscaling:updatepolicy:rollingupdate

为您环境的 Auto Scaling 组配置滚动更新。

命名空间: **`aws:autoscaling:updatepolicy:rollingupdate`**

名称	描述	默认	有效值
MaxBatchSize	滚动更新的每个批次中包含的实例数。	Auto Scaling 组的最小大小的三分之一（舍入到下一个最大整数）	1 到 10000
MinInstancesInService	终止其他实例时，Auto Scaling 组中必须处于运行中的最小实例数。	Auto Scaling 组的最小大小，或 Auto Scaling 组的最大大小减一（取二者中的较小值）。	0 到 9999

RollingUpdateEnabled	如果为 `true`，则为环境启用滚动更新。在您需要对 Elastic Beanstalk 软件应用程序进行频繁的少量更新，并且希望避免应用程序停机时，滚动更新很有用。
	
	将此值设置为 true 会自动启用 `MaxBatchSize` 、`MinInstancesInService` 和 `PauseTime` 选项。设置这些选项中的任何选项也会自动将 `RollingUpdateEnabled` 选项值设置为 `true`。将
		`false`	`true`
			
			`false`

名称	描述	默认	有效值
	此选项设置为 `false` 会禁用滚动更新。 **Note** 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。		

名称	描述	默认	有效值

RollingUpdateType	其中包括三种类型：基于时间的滚动更新、基于运行状况的滚动更新和不可变更新。
	
	基于时间的滚动更新在批次 PauseTime 之间应用。基于运行状况的滚动更新会等新实例通过运行状况检查后再继续下一个批次。不可变更新将启动新 Auto Scaling 组中的一整组实例。
	
	Note
	如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。
		`Time`	`Time`
			
			`Health`
			
			`Immutable`

名称	描述	默认	有效值
PauseTime	Elastic Beanstalk 服务在完成一批实例更新之后到开始下一批实例更新之前要等待的时长（以秒、分钟或小时为单位）。	基于实例类型和容器自动计算。	PT0S*（0 秒）至 PT1H（1 小时）
Timeout	在取消更新之前，等待一个实例批次中的所有实例通过运行状况检查的最长时间（以分钟或小时为单位）。	PT30M（30 分钟）	PT5M*（5 分钟）至 PT1H（1 小时） *ISO8601 持续时间格式：PT#H#M#S，其中每个 # 分别代表小时数、分钟数和/或秒数。

aws:ec2:instances

配置环境的实例，包括 Spot 选项。此命名空间是 `aws:autoscaling:launchconfiguration` 和 `aws:autoscaling:asg` 的补充。

有关更多信息，请参阅 the section called “Auto Scaling 组”。

命名空间: **`aws:ec2:instances`**

名称	描述	默认	有效值

EnableSpot	为您的环境启用 Spot 实例请求。如果为 `false`，此命名空间中的某些选项不会生效。	`false`	`true`
			
			`false`

InstanceTypes	您希望环境使用的实例类型的逗号分隔列表（例如，`t2.micro, t3.micro` ）。	两种实例类型	一至十种 EC2 实例类型。我们建议至少两种。 因账户、区域和可用区而异。您可以获取通过这些

名称	描述	默认	有效值
	当未激活 Spot 实例（EnableSpot 为 false）时，仅使用列表中的第一种实例类型。 此选项的列表中的第一个实例类型等同于 InstanceType 命名空间中的 aws:autoscaling:launchconfiguration 选项的值。我们不建议使用后一个选项，因为它已经过时了。如果您同时指定两者，则使用 InstanceTypes 选项的列表中的第一个实例类型，并忽略 InstanceType 。 可用的实例类型取决于使用的可用区和区域。如果您选择子网，则包含该子网的可用区将决定可用的实例类型。 • Elastic Beanstalk 不支持 Amazon EC2 Mac 实例类型。 • 有关 Amazon EC2 实例系列和类型的更多信息，请参阅 Amazon EC2 用户指南中的实例类型或亚马逊 EC2 用户指南中的实例类型。 • 有关各区域可用实例类型的更多信息，请参阅 Amazon EC2 用户指南中的可用实例类型或 Ama zon EC2 用户指南中的可用实例类型。 Note 一些较旧的 AWS 账户可能会为 Elastic Beanstalk 提供不支持竞价型实例的默认实例类型（例如 t1.micro）。如果激活	的列表。 因账户和区域而异。	值筛选的 Amazon EC2 实例类型的列表。有关更多信息，请参阅 Amazon EC2 用户指南中的可用实例类型或 A mazon EC2 用户指南中的可用实例类型。 实例类型必须均属于同一架构（arm64、x86_64、i386）。 Supported Architectures 也是此命名空间的一部分。如果您为 Supported Architectures 提供了任何值，则为 InstanceTypes 输入的值必须属于且仅属于为 Supported Architectures 提供的架构之一。

名称	描述	默认	有效值
	Spot 实例请求，并收到不支持 Spot 的实例类型的相关错误，请务必配置支持 Spot 的实例类型。要选择 Spot 实例类型，请使用 Spot Instance Advisor。 当您更新环境配置并从 `InstanceTypes` 选项中删除一个或多个实例类型时，Elastic Beanstalk 会终止在任何已删除的实例类型上运行的任何 Amazon EC2 实例。然后，您环境的 Auto Scaling 组根据需要启动新实例，以使用当前指定的实例类型来完成所需的容量。		
SpotFleet OnDemandBase	扩展环境时，在考虑 Spot 实例之前，Auto Scaling 组预配置的最小按需实例数。 此选项仅在 `EnableSpot` 为 `true` 时有意义。	`0`	`0` 命名空间中 `MaxSize` 至 `aws:autoscaling:asg` 选项
SpotFleet OnDemandAboveBasePercentage	Auto Scaling 组在 `SpotOnDemandBase` 实例之外作为额外容量预配置的按需实例的百分比。 此选项仅在 `EnableSpot` 为 `true` 时有意义。	`0` 适用于单实例环境 `70` 适用于负载均衡环境	`0` 到 `100`

名称	描述	默认	有效值

SpotMaxPrice	您愿意为 Spot 实例支付的每单位小时的最高价（USD）。有关竞价型实例最高价格选项的建议，请参阅 Amazon EC2 用户指南中的竞价型实例定价历史记录。
	
	此选项仅在 `EnableSpot` 为 `true` 时有意义。
		每种实例类型的按需价格。在这种情况下，该选项的值为 `null`。	`0.001` 到 `20.0`
			
			`null`

SupportedArchitectures	您希望环境使用的 EC2 实例架构类型的逗号分隔列表。
	
	Elastic Beanstalk 支持基于以下处理器架构的实例类型：
	
	• AWS Graviton 64 位 Arm 架构 (arm64)
	• 64 位架构 (x86_64)
	• 32 位架构 (i386)
	
	有关处理器架构和 Amazon EC2 实例类型的更多信息，请参阅 the section called “Amazon EC2 实例类型”。
		无	`arm64`
			
			`x86_64`
			
			`i386`
			
			Note
			大多数 Elastic Beanstalk 平台都不支持 32 位架构 `i386`。我们建议您选择 `x86_64` 或 `arm64` 架构类型代替。

aws:ec2:vpc

配置环境以在自定义 Amazon Virtual Private Cloud（Amazon VPC）中启动资源。如果您不在此命名空间中配置设置，Elastic Beanstalk 将在默认 VPC 中启动资源。

命名空间: **aws:ec2:vpc**

名称	描述	默认	有效值
VPCId	您的 Amazon VPC 的 ID。	无	
Subnets	一个或多个 Auto Scaling 组子网的 ID。如果您有多个子网，请将此值指定为子网 ID 的单个逗号分隔字符串（例如，`"subnet-11111111,subnet-22222222"`）。	无	
ELBSubnets	Elastic Load Balancer 的一个或多个子网的 ID。如果您有多个子网，请将此值指定为子网 ID 的单个逗号分隔字符串（例如，`"subnet-11111111,subnet-22222222"`）。	无	
ELBScheme	如果您要在 Amazon VPC 中创建一个内部负载均衡器，请指定 `internal`，从而确保不能从 Amazon VPC 的外部访问您的 Elastic Beanstalk 应用程序。如果指定 `public` 或 `internal` 以外的值，Elastic Beanstalk 会忽略此值。	`public`	`public` `internal`
DBSubnets	包含数据库子网的 ID。这仅可用于以下情况：需要将 Amazon RDS 数据库实例添加到应用程序中。如果您有多个子网，请将此值指定为子网 ID 的单个逗号分隔字符串（例如，`"subnet-11111111,subnet-22222222"`）。	无	
AssociatePublicIpAddress	指定是否在您的 Amazon VPC 中启动具有公有 IP 地址的实例。具有公有 IP 地址的实例无需 NAT 设备即可与 Internet 通信。如果要在单个公有子网中包含您的负载均衡器和实例，则必须将值设置为 `true`。 此选项对单实例环境没有影响，该环境始终具有带弹性 IP 地址的单个 Amazon EC2 实例。该选项与负载平衡、可扩展的环境相关。	无	`true` `false`

aws:elasticbeanstalk:application

为您的应用程序配置运行状况检查路径。有关更多信息，请参阅 基本运行状况报告。

命名空间: **aws:elasticbeanstalk:application**

名称	描述	默认	有效值
应用程序运行状况检查 URL	运行状况检查请求发送到的路径。如果未设置此路径，负载均衡器将尝试在端口 80 上建立一个 TCP 连接，以验证应用程序的运行状况状态。设置为以 / 开头的路径可将 HTTP GET 请求发送到该路径。您还可以在该路径前面包含协议 (HTTP、HTTPS、TCP 或 SSL) 和端口以检查 HTTPS 连接情况或使用非默认端口。 **Note** 如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。	无	有效值包括： / (HTTP GET to root path) */health* HTTPS:443/ HTTPS:443/ *health*

EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

aws:elasticbeanstalk:application:environment

为您的应用程序配置环境属性。

命名空间: **aws:elasticbeanstalk:application:environment**

名称	描述	默认	有效值
任意环境变量名称。	传入密钥-值对。	无	任意环境变量值。

请参阅环境属性和其他软件设置了解更多信息。

aws:elasticbeanstalk:cloudwatch:logs

为应用程序配置实例日志流式传输。

命名空间: **aws:elasticbeanstalk:cloudwatch:logs**

名称	描述	默认	有效值
StreamLogs	指定是否在 CloudWatch 日志中为代理和部署日志创建组，以及是否从环境中的每个实例流式传输日志。	false	true false
DeleteOnTerminate	指定是否在环境终止后删除日志组。如果为 false，则日志保留 RetentionInDays 天。	false	true false
RetentionInDays	日志事件在到期前保留的天数。	7	1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、3653

aws:elasticbeanstalk:cloudwatch:logs:health

为应用程序配置环境运行状况日志流式传输。

命名空间: **aws:elasticbeanstalk:cloudwatch:logs:health**

名称	描述	默认	有效值
HealthStreamingEnabled	对于启用了增强型运行状况报告的环境，指定是否在环境运行状况 CloudWatch 日志中创建群组并存档 Elastic Beanstalk 环境运行状况数据。有关启用增强型运行状况的信息，请参阅 aws:elasticbeanstalk:healthreporting:system。	false	true false

名称	描述	默认	有效值
DeleteOnTerminate	指定是否在终止环境后删除日志组。如果为 `false`，则运行状况数据将保留 `RetentionInDays` 天。	`false`	`true` `false`
RetentionInDays	在存档的运行状况数据过期前要保留其的天数。	7	1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、3653

aws:elasticbeanstalk:command

为您的应用程序代码配置部署策略。有关更多信息，请参阅 the section called “部署选项”。

命名空间: **`aws:elasticbeanstalk:command`**

名称	描述	默认	有效值

DeploymentPolicy	选择应用程序版本部署的部署策略。
	Note
	如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。
		`AllAtOnce`	`AllAtOnce`
			`Rolling`
			`RollingWithAdditionalBatch`
			`Immutable`
			`TrafficSplitting`

Timeout	等待实例完成执行命令的时间（单位：秒）。	`600`	1 到 `3600`

名称	描述	默认	有效值
	Elastic Beanstalk 内部向 `Timeout` 值添加 240 秒（4 分钟）。例如，默认的有效超时为 840 秒（600 + 240）或 14 分钟。		

BatchSizeType	中指定的数字类型BatchSize。
	
	Note
	如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。
		`Percentage`	`Percentage`
			
			`Fixed`

BatchSize	Auto Scaling 组中要同时执行部署的 Amazon EC2 实例的百分比或固定数量。有效值因使用的BatchSizeType设置而异。 **Note** 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。	`100`	1 到 `100` (`Percentage`)。 1到 aws:autoscaling: asg:: () MaxSize `Fixed`

IgnoreHealthCheck	不要由于运行状况检查失败而取消部署。	`false`	`true`
			
			`false`

aws:elasticbeanstalk:environment

配置您的环境的架构和服务角色。

命名空间: **aws:elasticbeanstalk:environment**

名称	描述	默认	有效值
EnvironmentType	设置为 SingleInstance 可启动一个 EC2 实例而无需负载均衡器。	LoadBalanced	SingleInstance LoadBalanced
ServiceRole	一个 IAM 角色的名称，该角色供 Elastic Beanstalk 用来管理环境的资源。指定角色名称（可以选择添加自定义路径作为前缀）或其 ARN。 示例： • aws-elasticbeanstalk-service-role • *custom-path* /*custom-role* • arn:aws:iam::123456789012:role/aws-elasticbeanstalk-service-role **Note** 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。	无	IAM 角色名称、路径/名称或 ARN
LoadBalancerType	用于环境的负载均衡器的类型。有关更多信息，请参阅 the section called “负载均衡器”。	classic	classic application network

名称	描述	默认	有效值

LoadBalancerIsShared	指定环境的负载均衡器是专用的还是共享的。只能为 Application Load Balancer 设置此选项。环境创建后无法更改。
	
	当为 `false` 时，环境具有自己的专用负载均衡器（由 Elastic Beanstalk 创建和管理）。当为 `true` 时，环境使用共享的负载均衡器，该负载均衡器由您创建并在 aws:elbv2:loadbalancer 命名空间的 `SharedLoadBalancer` 选项中指定。
		`false`	`true`
			
			`false`

aws:elasticbeanstalk:environment:process:default

配置您的环境的默认过程。

命名空间: **`aws:elasticbeanstalk:environment:process:default`**

名称	描述	默认	有效值
DeregistrationDelay	在取消注册之前等待活动请求完成的时间（单位：秒）。	`20`	`0` 到 `3600`
HealthCheckInterval	Elastic Load Balancing 检查应用程序的 Amazon EC2 实例运行状况的时间间隔（单位：秒）。	使用 Classic 或应用程序负载均衡器：`15` 使用网络负载均衡器：`30`	使用 Classic 或应用程序负载均衡器：`5` 至 `300` 使用网络负载均衡器：`10`、`30`
HealthCheckPath	运行状况检查的 HTTP 请求发送到的路径。	`/`	可路由路径。
HealthCheckTimeout	在运行状况检查期间等待响应的时间（单位：秒）。	`5`	`1` 到 `60`

名称	描述	默认	有效值
	此选项仅适用于使用应用程序负载均衡器的环境。		
HealthyThresholdCount	Elastic Load Balancing 更改实例运行状况状态前的连续成功请求数。	使用 Classic 或应用程序负载均衡器：3 使用网络负载均衡器：5	2 到 10
MatcherHTTPCode	指示实例正常的 HTTP 代码 (以逗号分隔) 的列表。 此选项仅适用于使用网络或应用程序负载均衡器的环境。	200	使用应用程序负载均衡器：200 至 499 使用网络负载均衡器：200 至 399
Port	进程侦听的端口。	80	1 到 65535
Protocol	进程使用的协议。 使用应用程序负载均衡器时，您只能将此选项设置为 HTTP 或 HTTPS。 使用网络负载均衡器时，您只能将此选项设置为 TCP。	使用 Classic 或应用程序负载均衡器：HTTP 使用网络负载均衡器：TCP	TCP HTTP HTTPS
StickinessEnabled	设置为 true 可启用粘性会话。 此选项仅适用于使用应用程序负载均衡器的环境。	'false'	'false' 'true'

名称	描述	默认	有效值
StickinessLBCookieDuration	粘性会话 Cookie 的生存期（单位：秒） 此选项仅适用于使用应用程序负载均衡器的环境。	86400 (一天)	1 到 604800
StickinessType	设置为 lb_cookie 可以为粘性会话使用 Cookie。 此选项仅适用于使用应用程序负载均衡器的环境。	lb_cookie	lb_cookie
UnhealthyThresholdCount	Elastic Load Balancing 更改实例运行状况状态前的连续失败请求数。	5	2 到 10

aws:elasticbeanstalk:environment:process:process_name

配置您的环境的其他过程。

命名空间: **aws:elasticbeanstalk:environment:process:*process_name***

名称	描述	默认	有效值
DeregistrationDelay	在取消注册之前等待活动请求完成的时间（单位：秒）。	20	0 到 3600
HealthCheckInterval	Elastic Load Balancing 检查应用程序的 Amazon EC2 实例运行状况的时间间隔（单位：秒）。	使用 Classic 或应用程序负载均衡器：15 使用网络负载均衡器：30	使用 Classic 或应用程序负载均衡器：5 至 300

名称	描述	默认	有效值
			使用网络负载均衡器：10、30
HealthCheckPath	运行状况检查的 HTTP 请求发送到的路径。	/	可路由路径。
HealthCheckTimeout	在运行状况检查期间等待响应的时间（单位：秒）。 此选项仅适用于使用应用程序负载均衡器的环境。	5	1 到 60
HealthyThresholdCount	Elastic Load Balancing 更改实例运行状况状态前的连续成功请求数。	使用 Classic 或应用程序负载均衡器：3 使用网络负载均衡器：5	2 到 10
MatcherHTTPCode	指示实例正常的 HTTP 代码（以逗号分隔）的列表。 此选项仅适用于使用网络或应用程序负载均衡器的环境。	200	使用应用程序负载均衡器：200 至 499 使用网络负载均衡器：200 至 399
Port	进程侦听的端口。	80	1 到 65535

名称	描述	默认	有效值
Protocol	进程使用的协议。 使用应用程序负载均衡器时，您只能将此选项设置为 HTTP 或 HTTPS。 使用网络负载均衡器时，您只能将此选项设置为 TCP。	使用 Classic 或应用程序负载均衡器：HTTP 使用网络负载均衡器：TCP	TCP HTTP HTTPS
StickinessEnabled	设置为 true 可启用粘性会话。 此选项仅适用于使用应用程序负载均衡器的环境。	'false'	'false' 'true'
StickinessLBCookieDuration	粘性会话 Cookie 的生存期（单位：秒） 此选项仅适用于使用应用程序负载均衡器的环境。	86400 (一天)	1 到 604800
StickinessType	设置为 lb_cookie 可以为粘性会话使用 Cookie。 此选项仅适用于使用应用程序负载均衡器的环境。	lb_cookie	lb_cookie
UnhealthyThresholdCount	Elastic Load Balancing 更改实例运行状况状态前的连续失败请求数。	5	2 到 10

aws:elasticbeanstalk:environment:proxy:staticfiles

您可以使用以下命名空间来配置代理服务器提供静态文件。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。这将减少您的应用程序必须处理的请求的数量。

在源代码中将代理服务器提供的路径映射到包含静态资产的文件夹。在此命名空间中定义的每个选项都映射不同的路径。

> **Note**
>
> 此命名空间适用于基于 Amazon Linux 2 的平台分支。如果您的环境使用基于 Amazon Linux AMI（在 Amazon Linux 2 之前）的平台版本，请参阅the section called “特定于平台的选项”以获取特定于平台的静态文件命名空间。

命名空间: **`aws:elasticbeanstalk:environment:proxy:staticfiles`**

名称	值
代理服务器提供文件的路径。值以 `/` 开始。 例如，指定 `/images` 以在 *`subdomain`*`.elasticbeanstalk.com/images` 提供文件。	包含文件的文件夹名称。 例如，指定 `staticimages` 以在源代码包顶层从名为 `staticimages` 的文件夹中提供文件。

aws:elasticbeanstalk:healthreporting:system

为您的环境配置增强型运行状况报告。

命名空间: **`aws:elasticbeanstalk:healthreporting:system`**

名称	描述	默认	有效值

SystemType	运行状况报告系统（基本或增强）。增强型运行状况报告需要一个服务角色和一个版本 2 或更高的平台版本。	`basic`	`basic`
			
			`enhanced`

名称	描述	默认	有效值
	Note 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。		
ConfigDocument	一个 JSON 文档，描述要发布到的环境和实例指标 CloudWatch。	无	
EnhancedHealthAuthEnabled	为内部 API 启用授权，Elastic Beanstalk 使用此授权将增强型运行状况信息从您的环境实例传达到 Elastic Beanstalk 服务。 有关更多信息，请参阅 the section called “增强型运行状况角色”。 **Note** 此选项仅适用于增强型运行状况报告（例如 `SystemType` 设置为 `enhanced` 时）。	`true`	`true` `false`

HealthCheckSuccessThreshold	降低阈值以便实例能够通过运行状况检查。
	Note
	如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。
		`Ok`	`Ok`
			`Warning`
			`Degraded`
			`Severe`

aws:elasticbeanstalk:hostmanager

在环境中配置 EC2 实例以将轮换日志上传到 Amazon S3。

命名空间: **aws:elasticbeanstalk:hostmanager**

名称	描述	默认	有效值
LogPublicationControl	将应用程序的 Amazon EC2 实例日志文件复制到与应用程序相关联的 Amazon S3 存储桶。	false	true false

aws:elasticbeanstalk:managedactions

为您的环境配置托管平台更新。

命名空间: **aws:elasticbeanstalk:managedactions**

名称	描述	默认	有效值
ManagedActionsEnabled	启用托管平台更新。 在将此项设置为 true 时，还必须指定 PreferredStartTime 和 UpdateLevel。	false	true false
PreferredStartTime	配置托管操作的维护时段 (采用 UTC 表示)。 例如，"Tue:09:00"。	无	日期和时间 *day*:*hour*:*minute* 格式的日期和时间。
ServiceRoleForManagedUpdates	Elastic Beanstalk 用于为您的环境执行托管平台更新的 IAM 角色的名称。 您可以使用为 ServiceRole 命名空间的 aws:elasticbeanstalk:environment 选项指定同一个角色，也可以使用账户的托管更新服务相关角色。在后一种情况下，如	无	与 ServiceRole 相同 或 AWSServiceRoleForElasticBeanstalkMan

名称	描述	默认	有效值
	果账户还没有托管更新服务相关角色，Elastic Beanstalk 会创建该角色。		agedUpdates

aws:elasticbeanstalk:managedactions:platformupdate

为您的环境配置托管平台更新。

命名空间: **aws:elasticbeanstalk:managedactions:platformupdate**

名称	描述	默认	有效值
UpdateLevel	要通过托管平台更新应用的最高级别的更新。平台版本的格式为 *major.minor.patch*。例如，在 2.0.8 中，主版本为 2，次版本为 0，修补版本为 8。	无	patch 仅适用于修补版本更新。 minor 适用于次版本更新和修补版本更新。
InstanceRefreshEnabled	启用每周实例替换。 需要将 ManagedActionsEnabled 设置为 true。	false	true false

aws:elasticbeanstalk:monitoring

配置您的环境以终止运行状况检查失败的 EC2 实例。

命名空间: **aws:elasticbeanstalk:monitoring**

名称	描述	默认	有效值
Automatically Terminate Unhealthy Instances	如果实例无法通过运行状况检查，请将其终止。	true	true

名称	描述	默认	有效值
	Note 此选项仅在早期环境上受支持。它根据能够达到实例的运行状况及其他基于实例的指标确定了实例的运行状况。Elastic Beanstalk 不提供根据应用程序运行状况自动终止实例的方法。		`false`

aws:elasticbeanstalk:sns:topics

为您的环境配置通知。

命名空间: **`aws:elasticbeanstalk:sns:topics`**

名称	描述	默认	有效值
Notification Endpoint	在该端点处，系统会通知您对应用程序产生影响的重要事件。 Note 如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。	无	
Notification Protocol	用于向您的端点发送通知的协议。	`email`	`http` `https`

名称	描述	默认	有效值

			`email`
			
			`email-json`
			
			`sqs`

Notification Topic ARN	已订阅主题的 Amazon Resource Name (ARN)。	无	
Notification Topic Name	已订阅主题的名称。	无	

aws:elasticbeanstalk:sqsd

为工作线程环境配置 Amazon SQS 队列。

命名空间: **`aws:elasticbeanstalk:sqsd`**

名称	描述	默认	有效值
WorkerQueueURL	队列的 URL，工作线程环境层中的守护程序从该队列读取消息。 **Note** 当您不指定值时，Elastic Beanstalk 自动创建的队列是标准 Amazon SQS 队列。当您提供值时，可以提供标准或 FIFO Amazon SQS 队列的 URL。请注意，如果提供 FIFO 队列，不支持定期任务。	自动生成	如果您不指定值，则 Elastic Beanstalk 会自动创建队列。
HttpPath	将 HTTP POST 消息发送到的应用程序的相对路径。	/	

名称	描述	默认	有效值

MimeType	HTTP POST 请求中发送的消息的 MIME 类型。	`application/json`	`application/json`
			`application/x-www-form-urlencoded`
			`application/xml`
			`text/plain`
			自定义 MIME 类型。

HttpConnections	与 Amazon EC2 实例中任何应用程序之间的最大并发连接数。 **Note** 如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。	`50`	1 到 `100`
ConnectTimeout	等待成功连接到应用程序的时长（单位：秒）。	`5`	1 到 `60`
InactivityTimeout	在与应用程序的现有连接上等待响应的时长（单位：秒）。 消息会重新处理，直到守护程序从工作线程环境层中的应用程序收到 200 (OK) 响应或 `Retention Period` 过期。	`299`	1 到 `36000`

名称	描述	默认	有效值
VisibilityTimeout	锁定来自 Amazon SQS 队列的入站消息以进行处理的时长（以秒为单位）。在配置的时长之后，再次使消息在队列中可见，以供任何其他守护程序读取。	300	0 到 43200
ErrorVisibilityTimeout	在处理尝试由于显式错误而失败后，Elastic Beanstalk 将消息返回到 Amazon SQS 队列之前经过的时长（以秒为单位）。	2 秒	0 至 43200 秒
RetentionPeriod	消息有效和等待主动处理的时长（单位：秒）	345600	60 到 1209600
MaxRetries	Elastic Beanstalk 在将消息移动到死信队列之前，尝试向处理消息的 Web 应用程序发送消息的最大尝试次数。	10	1 到 100

aws:elasticbeanstalk:trafficsplitting

为您的环境配置流量拆分部署。

当您将 aws:elasticbeanstalk:command 命名空间的 DeploymentPolicy 选项设置为 TrafficSplitting 时，将应用此命名空间。有关部署策略的更多信息，请参阅the section called “部署选项”。

命名空间: **aws:elasticbeanstalk:trafficsplitting**

名称	描述	默认	有效值
NewVersionPercent	对于运行您正在部署的新应用程序版本的环境实例，Elastic Beanstalk 转移到这些实例的传入客户端流量的初始百分比。	10	1 到 100

名称	描述	默认	有效值
EvaluationTime	在初始正常部署之后，Elastic Beanstalk 等待的时间段（以分钟为单位），在经过该时间后，会继续将所有传入的客户端流量转移到正在部署的新应用程序版本。	5	3 到 600

aws:elasticbeanstalk:xray

运行 AWS X-Ray 守护程序以中继来自 X-Ray 集成应用程序的跟踪信息。

命名空间: **`aws:elasticbeanstalk:xray`**

名称	描述	默认	有效值
`XRayEnabled`	设为 `true` 可在环境中的实例上运行 X-Ray 守护程序。	`false`	`true` `false`

aws:elb:healthcheck

为经典负载均衡器配置运行状况检查。

命名空间: **`aws:elb:healthcheck`**

名称	描述	默认	有效值
HealthyThreshold	Elastic Load Balancing 更改实例运行状况状态前的连续成功请求数。	3	2 到 10
Interval	Elastic Load Balancing 检查应用程序的 Amazon EC2 实例运行状况的时间间隔。	10	5 到 300
Timeout	Elastic Load Balancing 在将实例视为无响应之前等待的时间（单位：秒）	5	2 到 60
UnhealthyThreshold	Elastic Load Balancing 更改实例运行状况状态前的连续失败请求数。	5	2 到 10

名称	描述	默认	有效值
（已弃用）Target	运行状况检查发送到的后端实例上的目标。请改为在 Application Healthcheck URL 命名空间中使用 aws:elasticbeanstalk:application 。	TCP:80	采用格式 *PROTOCOL*:*PORT*/*PATH* 的目标

aws:elb:loadbalancer

配置您环境的经典负载均衡器。

此命名空间中的多个选项已不再受支持，以支持 aws:elb:listener 命名空间中特定于侦听器的选项。使用这些不再受支持的选项，您只能在标准端口上配置两个侦听器（一个安全一个不安全）。

命名空间: **aws:elb:loadbalancer**

名称	描述	默认值	有效值
CrossZone	将负载均衡器配置为在所有可用区中的所有实例间 (而不是仅在每个区域中) 均匀地路由流量。 **Note** 如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。	false	true false
SecurityGroups	将您创建的一个或多个安全组分配到负载均衡器。	无	一个或多个安全组 ID。
ManagedSecurityGroup	将现有安全组分配给环境的负载均衡器，而不是创建一个新安全组。要使用此设置，请更新此命名空间中的 SecurityGroups 设置以包含安全组的 ID，然后删除自动创建的安全组 ID（如果已创建）。	无	安全组 ID。

名称	描述	默认值	有效值
	为了允许从负载均衡器到环境的 EC2 实例的流量，Elastic Beanstalk 会向实例的安全组添加一条规则，允许来自托管安全组的入站流量。		
（已弃用）LoadBalancerHTTPPort	不安全的侦听器要侦听的端口。	80	OFF 80
（已弃用）LoadBalancerPortProtocol	不安全的侦听器上使用的协议。	HTTP	HTTP TCP
（已弃用）LoadBalancerHTTPSPort	安全的侦听器要侦听的端口。	OFF	OFF 443 8443
（已弃用）LoadBalancerSSLPortProtocol	安全的侦听器上使用的协议。	HTTPS	HTTPS SSL
（已弃用）SSLCertificateId	要绑定到安全的侦听器的 SSL 证书的 Amazon Resource Name (ARN)。	无	

aws:elb:listener

在经典负载均衡器上配置默认侦听器（端口 80）。

命名空间: **`aws:elb:listener`**

名称	描述	默认值	有效值
ListenerProtocol	侦听器使用的协议。	HTTP	HTTP TCP
InstancePort	此侦听器用于与 EC2 实例通信的端口。	80	1 到 65535

名称	描述	默认值	有效值
InstanceProtocol	此侦听器用于与 EC2 实例通信的协议。 它必须位于与 ListenerProtocol 相同的 Internet 协议层中。其安全级别也必须与使用与该侦听器相同的 InstancePort 的任何其他侦听器相同。 例如，如果 ListenerProtocol 是 HTTPS (应用程序层，使用安全连接)，您可以将 InstanceProtocol 设置为 HTTP (也位于应用程序层，使用不安全的连接)。此外，如果将 InstancePort 设置为 80，您必须在将 InstanceProtocol 设置为 HTTP 的所有其他侦听器中将 InstancePort 设置为 80。	HTTP (当 ListenerProtocol 为 HTTP 时) TCP (当 ListenerProtocol 为 TCP 时)	HTTP 或 HTTPS 当 ListenerProtocol 为 HTTP 或 HTTPS TCP 或 SSL 当 ListenerProtocol 为 TCP 或 SSL
PolicyNames	应用于该侦听器的端口的策略名称的逗号分隔列表。我们建议您改用aws:elb:policies命名空间 LoadBalancerPorts 选项。	无	
ListenerEnabled	指定是否启用该侦听器。如果指定 false，则此侦听器不包含在负载均衡器中。	true	true false

aws:elb:listener:listener_port

在经典负载均衡器上配置其他侦听器。

命名空间: **aws:elb:listener:*listener_port***

名称	描述	默认值	有效值
ListenerProtocol	侦听器使用的协议。	HTTP	HTTP HTTPS TCP SSL

名称	描述	默认值	有效值
InstancePort	此侦听器用于与 EC2 实例通信的端口。	与 *listener_port* 相同。	1 到 65535
InstanceProtocol	此侦听器用于与 EC2 实例通信的协议。 它必须位于与 ListenerProtocol 相同的 Internet 协议层中。其安全级别也必须与使用与该侦听器相同的 InstancePort 的任何其他侦听器相同。 例如，如果 ListenerProtocol 是 HTTPS (应用程序层，使用安全连接)，您可以将 InstanceProtocol 设置为 HTTP (也位于应用程序层，使用不安全的连接)。此外，如果将 InstancePort 设置为 80，您必须在将 InstanceProtocol 设置为 HTTP 的所有其他侦听器中将 InstancePort 设置为 80。	HTTP 当 ListenerProtocol 为 HTTP 或 HTTPS TCP 当 ListenerProtocol 为 TCP 或 SSL	HTTP 或 HTTPS 当 ListenerProtocol 为 HTTP 或 HTTPS TCP 或 SSL 当 ListenerProtocol 为 TCP 或 SSL
PolicyNames	应用于该侦听器的端口的策略名称的逗号分隔列表。我们建议您改用aws:elb:policies命名空间的 LoadBalancerPorts 选项。	无	
SSLCertificateId	要绑定到侦听器的 SSL 证书的 Amazon Resource Name (ARN)。	无	
ListenerEnabled	指定是否启用该侦听器。如果指定 false，则此侦听器不包含在负载均衡器中。	如果设置了任何其他选项，则为 true；否则为 false。	true false

aws:elb:policies

修改经典负载均衡器的默认粘性和全局负载均衡器策略。

命名空间: **`aws:elb:policies`**

名称	描述	默认	有效值

ConnectionDrainingEnabled	指定负载均衡器是否维持与已运行状况不佳或取消注册的实例之间的现有连接以完成正在进行的请求。
	
	Note
	如果您使用 Elastic Beanstalk 控制台或 EB CLI 创建环境，则无法在配置文件中设置此选项。控制台和 EB CLI 使用建议的值覆盖此选项。
		`false`	`true`
			
			`false`

ConnectionDrainingTimeout	在强制关闭连接之前，负载均衡器在连接耗尽过程中维持与实例之间的现有连接的最大秒数。 **Note** 如果您使用 Elastic Beanstalk 控制台创建环境，则无法在配置文件中设置此选项。控制台使用建议的值覆盖此选项。	`20`	1 到 3600
ConnectionSettingIdleTimeout	负载均衡器等待通过连接发送或接收任何数据的时间（单位：秒）。如果此时段后未发送或接收任何数据，则负载均衡器将关闭连接。	`60`	1 到 3600
LoadBalancerPorts	应用默认策略 (`AWSEB-ELB-StickinessPolicy`) 的侦听器端口的逗号分隔列表。	无	您可以使用 `:all` 指示所有侦听器端口

名称	描述	默认	有效值
Stickiness Cookie Expiration	每个 Cookie 的有效时间，以秒为单位。使用默认策略 (`AWSEB-ELB-StickinessPolicy`)。	`0`	`0` 到 `1000000`
Stickiness Policy	将一个用户会话绑定到某一特定的服务器实例，以便系统将用户在此会话期间发出的所有请求都发送到同一服务器实例。使用默认策略 (`AWSEB-ELB-StickinessPolicy`)。	`false`	`true false`

aws:elb:policies:policy_name

为经典负载均衡器创建其他负载均衡器策略。

命名空间: **`aws:elb:policies:`*`policy_name`***

名称	描述	默认	有效值
CookieName	应用程序生成的 Cookie 的名称，用于控制 `AppCookieStickinessPolicyType` 策略的会话生存期。此策略只能与 HTTP/HTTPS 侦听器关联。	无	
InstancePorts	应用此策略的实例端口的逗号分隔列表。	无	端口列表或 `:all`
LoadBalancerPorts	应用此策略的侦听器端口的逗号分隔列表。	无	端口列表或 `:all`
ProxyProtocol	对于 `ProxyProtocolPolicyType` 策略，指定是否包含 TCP 消息的原始请求的 IP 地址和端口。此策略只能与 TCP/SSL 侦听器关联。	无	`true false`
PublicKey	对后端服务器进行身份验证时要使用的 `PublicKeyPolicyType` 策略的公有密钥的内容。此策略不能直接应用于后端	无	

名称	描述	默认	有效值
	服务器或侦听器。必须作为 BackendServerAuthenticationPolicyType 策略的一部分。		
PublicKeyPolicyNames	控制后端服务器身份验证的 PublicKeyPolicyType 策略的策略名称（来自 BackendServerAuthenticationPolicyType 策略）的逗号分隔列表。此策略只能与使用 HTTPS/SSL 的后端服务器关联。	无	
SSLProtocols	对 SSLNegotiationPolicyType 策略（用于定义负载均衡器所接受的密码和协议）启用的 SSL 协议的逗号分隔列表。此策略只能与 HTTPS/SSL 侦听器关联。	无	
SSLReferencePolicy	符合安全最佳实践的预定义 AWS 安全策略的名称，您要为定义负载均衡器接受的密码和协议的SSLNegotiationPolicyType 策略激活该策略。此策略只能与 HTTPS/SSL 侦听器关联。	无	
Stickiness Cookie Expiration	每个 Cookie 的有效时间，以秒为单位。	0	0 到 1000000
Stickiness Policy	将一个用户会话绑定到某一特定的服务器实例，以便系统将用户在此会话期间发出的所有请求都发送到同一服务器实例。	false	true false

aws:elbv2:listener:default

在 Application Load Balancer 或 Network Load Balancer 上配置默认侦听器（端口 80）。

此命名空间不适用于使用共享负载均衡器的环境。共享的负载均衡器没有默认侦听器。

命名空间: **`aws:elbv2:listener:default`**

名称	描述	默认	有效值
DefaultProcess	无规则匹配时将流量转发到的流程的名称。	`default`	过程名称。

ListenerEnabled	设置为 `false` 可禁用侦听器。您可以使用此选项对端口 80 禁用默认侦听器。	`true`	`true`
			
			`false`

Protocol	要处理的流量的协议。	使用应用程序负载均衡器：HTTP 使用网络负载均衡器：TCP	使用应用程序负载均衡器：HTTP、HTTPS 使用网络负载均衡器：TCP
Rules	应用于侦听器的规则的列表 此选项仅适用于带 Application Load Balancer 的环境。	无	逗号分隔的规则名称列表。
SSLCertificateArns	要绑定到侦听器的 SSL 证书的 Amazon Resource Name (ARN)。 此选项仅适用于带 Application Load Balancer 的环境。	无	存储在 IAM 或 ACM 中的证书的 ARN。
SSLPolicy	指定要应用于监听器的安全策略。	无 (ELB 默认值)	负载均衡器安全策略的名称。

名称	描述	默认	有效值
	此选项仅适用于带 Application Load Balancer 的环境。		

aws:elbv2:listener:listener_port

在 Application Load Balancer 或 Network Load Balancer 上配置其他侦听器。

> **Note**
>
> 对于共享 Application Load Balancer，您只能指定 `Rule` 选项。其他选项不适用于共享的负载均衡器。

命名空间: **`aws:elbv2:listener:`*`listener_port`***

名称	描述	默认	有效值
DefaultProcess	无规则匹配时将流量转发到的流程的名称。	`default`	过程名称。

ListenerEnabled	设置为 `false` 可禁用侦听器。您可以使用此选项对端口 80 禁用默认侦听器。	`true`	`true`
			
			`false`

Protocol	要处理的流量的协议。	使用应用程序负载均衡器：`HTTP` 使用网络负载均衡器：`TCP`	使用应用程序负载均衡器：`HTTP`、`HTTPS` 使用网络负载均衡器：`TCP`
Rules	要应用于侦听器的规则的列表	无	逗号分隔的规则名称列表。

名称	描述	默认	有效值
	此选项仅适用于带 Application Load Balancer 的环境。 如果您的环境使用共享的 Application Load Balancer，并且您没有为任何侦听器指定此选项，则 Elastic Beanstalk 自动将 `default` 规则与端口 80 侦听器关联。		
SSLCertificateArns	要绑定到侦听器的 SSL 证书的 Amazon Resource Name (ARN)。 此选项仅适用于带 Application Load Balancer 的环境。	无	存储在 IAM 或 ACM 中的证书的 ARN。
SSLPolicy	指定要应用于监听器的安全策略。 此选项仅适用于带 Application Load Balancer 的环境。	无 (ELB 默认值)	负载均衡器安全策略的名称。

aws:elbv2:listenerrule:rule_name

为 Application Load Balancer 定义侦听器规则。如果请求与规则中的主机名或路径匹配，则负载均衡器将请求转发到指定的进程。要使用规则，请使用 `Rules` 命名空间中的 `aws:elbv2:listener:listener_port` 选项将其添加到侦听器中。

> **Note**
>
> 此命名空间不适用于使用网络负载均衡器的环境。

命名空间: **`aws:elbv2:listenerrule:`*`rule_name`***

名称	描述	默认	有效值
HostHeaders	要匹配的主机名列表。例如，`my.example.com` 。	专用负载均衡器：无 共享的负载均衡器：环境的 CNAME	每个名称最多可以包含 128 个字符。模式可包含大写和小写字母、数字、连字符 (–) 以及最多三个通配符（* 匹配零个或多个字符；？恰好匹配一个字符）。您可以列出多个名称，每个名称用逗号分隔。Application Load Balancer 最多支持五个 `HostHeader` 和 `PathPattern` 规则组合。 有关更多信息，请参阅 Application Load Balancer 用户指南中的主机条件。
PathPatterns	要匹配的路径模式（例如，`/img/*`）。 此选项仅适用于使用应用程序负载均衡器的环境。	无	每个模式最多可包含 128 个字符。模式可包含大写和小写字母、数字、连字符 (–) 以及最多三个通配符（* 匹配零个或多个字符；？恰好匹配一个字符）。您可以添加多个逗号分隔的路径模

名称	描述	默认	有效值
			式。Application Load Balancer 最多支持五个 `HostHeader` 和 `PathPattern` 规则组合。 有关更多信息，请参阅 Application Load Balancer 用户指南中的路径条件。
Priority	多个规则匹配时此规则的优先顺序。较小的数字优先。任何两个规则的优先级不能相同。 利用共享的负载均衡器，Elastic Beanstalk 将规则优先级视为跨共享环境的相对优先级，并在创建过程中将其映射到绝对优先级。	`1`	`1` 到 `1000`
Process	此规则与请求匹配时要将流量转发到的流程的名称。	`default`	过程名称。

aws:elbv2:loadbalancer

配置 Application Load Balancer。

对于共享的负载均衡器，只有 `SharedLoadBalancer` 和 `SecurityGroups` 选项有效。

> **Note**
>
> 此命名空间不适用于带 Network Load Balancer 的环境。

命名空间: **`aws:elbv2:loadbalancer`**

名称	描述	默认	有效值
AccessLogsS3Bucket	存储访问日志的 Amazon S3 存储桶。存储桶必须与环境处于同一区域并授予负载均衡器的写入访问权限。	无	存储桶名称。

AccessLogsS3Enabled	启用访问日志存储。	`false`	`true`
			
			`false`

AccessLogsS3Prefix	放在访问日志名称前的前缀。默认情况下，负载均衡器会将日志上传到您指定的存储桶 AWSLogs 中名为的目录。指定前缀以将该 AWSLogs 目录放在另一个目录中。	无	
IdleTimeout	在关闭到客户端和实例的连接之前，等待请求完成的时间（单位：秒）。	无	1 到 `3600`
ManagedSecurityGroup	将现有安全组分配给环境的负载均衡器，而不是创建一个新安全组。要使用此设置，请更新此命名空间中的 `SecurityGroups` 设置以包括安全组的 ID，并删除自动创建的安全组 ID (如果存在)。 为了允许从负载均衡器到环境的 EC2 实例的流量，Elastic Beanstalk 会向实例的安全组添加一条规则，允许来自托管安全组的入站流量。	Elastic Beanstalk 为您的负载均衡器闯进的安全组。	安全组 ID。
SecurityGroups	附加到负载均衡器的安全组的列表。 对于共享的负载均衡器，如果您未指定此值，则 Elastic Beanstalk 将检查其管理的现有安全组是否已连接到负载均衡器。如果没有连接到负载均衡器，Elastic Beanstalk 将创建一个	Elastic Beanstalk 为您的负载均衡器创建	安全组 ID 的逗号分隔列表。

名称	描述	默认	有效值
	安全组并将其附加到负载均衡器。当最后一个共享负载均衡器的环境终止时，Elastic Beanstalk 将删除此安全组。 负载均衡器安全组用于设置 Amazon EC2 实例安全组入口规则。	的安全组。	

名称	描述	默认	有效值
SharedLoadBalancer	共享负载均衡器的 Amazon Resource Name (ARN)。此选项仅与 Application Load Balancer 相关。当 aws:elasticbeanstalk:environment 命名空间的 `LoadBalancerIsShared` 选项设置为 `true` 时，这是必需的。创建环境后，您将无法更改共享的负载均衡器 ARN。 有效值的条件： • 它必须是环境所在 AWS 区域中有效的活跃负载均衡器。 • 它必须与环境位于同一 Amazon Virtual Private Cloud (Amazon VPC) 中。 • 它不能是由 Elastic Beanstalk 创建为另一环境的专用负载均衡器的负载均衡器。您可以使用前缀 `awseb-` 标识这些专用负载均衡器。 例如： `arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/FrontEndLB/0dbf78d8ad96abbc`	无	符合此处所述全部条件的有效负载均衡器的 ARN。

aws:rds:dbinstance

配置连接的 Amazon RDS 数据库实例。

命名空间: **`aws:rds:dbinstance`**

名称	描述	默认	有效值
DBAllocatedStorage	分配的数据库存储大小 (以吉字节为单位来指定)。	MySQL：5 Oracle：10 sqlserver-se：200 sqlserver-ex：30 sqlserver-web：30	MySQL：5-1024 Oracle：10-1024 sqlserver：无法修改
DBDeletionPolicy	指定在环境终止时保留、删除或创建数据库实例还是为数据库实例拍摄快照。 此选项与 HasCoupledDatabase 结合使用，也是此命名空间的一个选项。 **Warning** 删除数据库实例会导致数据永久丢失。	Delete	Delete Retain Snapshot
DBEngine	要用于此实例的数据库引擎的名称。	mysql	mysql oracle-se1 sqlserver-ex sqlserver-web sqlserver-se

名称	描述	默认	有效值
			`postgres`
DBEngineVersion	数据库引擎的版本号。	`5.5`	
DBInstanceClass	数据库实例类型。	`db.t2.micro` （对于并未在 Amazon VPC 中运行的环境，为 `db.m1.large` ）	有关更多信息，请参阅 Amazon Relational Database Service 用户指南中的数据库实例类。
DBPassword	数据库实例的主用户密码。	无	
DBSnapshotIdentifier	要用来恢复数据库的数据库快照的标识符。	无	
DBUser	数据库实例的主用户名称。	ebroot	

名称	描述	默认	有效值

HasCoupledDatabase	指定数据库实例是否与环境耦合。如果切换为 `true`，Elastic Beanstalk 会创建一个与您的环境耦合的新数据库实例。如果切换为 `false`，Elastic Beanstalk 开始将数据库实例与您的环境解耦。
	
	此选项与 `DBDeletionPolicy` 结合使用，也是此命名空间的一个选项。
	
	Note
	注意：如果在解耦前一个数据库之后将此值切换回 `true` ，Elastic Beanstalk 将使用前一个数据库选项设置创建一个新的数据库。但是，为了维护环境的安全性，它不会保留现有的 `DBUser` 和 `DBPassword` 设置。您需要再次指定 `DBUser` 和 `DBPassword` 。
		`false`	`true`
			
			`false`

MultiAZDatabase	指定是否需要创建数据库实例的多可用区部署。有关使用 Amazon Relational Database Service (RDS) 进行多可用区部署的更多信息，请参阅 Amazon Relational Database Service 用户指南中的区域和可用区。	`false`	`true`
			
			`false`

特定于平台的选项

某些 Elastic Beanstalk 平台定义特定于平台的选项命名空间。下面列出了每个平台的这些命名空间及其选项。

Note

以前，在基于 Amazon Linux AMI（在 Amazon Linux 2 之前）的平台版本中，以下两个功能及其各自的命名空间被视为特定于平台的功能，并且在此处按平台列出：

- 静态文件的代理配置 - `aws:elasticbeanstalk:environment:proxy:staticfiles`
- AWS X-Ray 支持 – `aws:elasticbeanstalk:xray`

在 Amazon Linux 2 平台版本中，Elastic Beanstalk 以一致的方式在所有支持平台上实现这些功能。相关的命名空间现在列在 the section called “常规选项” 页面中。对于命名空间名称不同的平台，我们只在此页面上提到它们。

平台

- Docker 平台选项
- Go 平台选项
- Java SE 平台选项
- 具有 Tomcat 的 Java 平台选项
- Linux 上的 .NET Core 平台选项
- .NET 平台选项
- Node.js 平台选项
- PHP 平台选项
- Python 平台选项
- Ruby 平台选项

Docker 平台选项

以下 Docker 特定的配置选项适用于 Docker 和预配置的 Docker 平台。

> **Note**
>
> 这些配置选项不适用于
>
> - 带 Docker Compose 的 Docker 平台 (Amazon Linux 2)
> - 多容器 Docker 平台 (Amazon Linux AMI)

命名空间: **aws:elasticbeanstalk:environment:proxy**

名称	描述	默认值	有效值
ProxyServer	指定要用作代理的 Web 服务器。	nginx	nginx none - Amazon Linux AM 和仅限 Docker（含 DC）

Go 平台选项

Amazon Linux AMI（先前的 Amazon Linux 2）平台选项

命名空间: **aws:elasticbeanstalk:container:golang:staticfiles**

您可以使用以下命名空间来配置代理服务器提供静态文件。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。这将减少您的应用程序必须处理的请求的数量。

在源代码中将代理服务器提供的路径映射到包含静态资产的文件夹。在此命名空间中定义的每个选项都映射不同的路径。

名称	值
代理服务器将提供文件的路径。 示例：/images 将在 *subdomain* .elasticbeanstalk.com/images 提供文件。	包含文件的文件夹名称。 示例：staticimages 将在源代码包顶层从名为 staticimages 的文件夹中提供文件。

Java SE 平台选项

Amazon Linux AMI（先前的 Amazon Linux 2）平台选项

命名空间: **aws:elasticbeanstalk:container:java:staticfiles**

您可以使用以下命名空间来配置代理服务器提供静态文件。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。这将减少您的应用程序必须处理的请求的数量。

在源代码中将代理服务器提供的路径映射到包含静态资产的文件夹。在此命名空间中定义的每个选项都映射不同的路径。

名称	值
代理服务器将提供文件的路径。 示例：`/images` 将在 *subdomain* `.elasticbeanstalk.com/images` 提供文件。	包含文件的文件夹名称。 示例：`staticimages` 将在源代码包顶层从名为 `staticimages` 的文件夹中提供文件。

具有 Tomcat 的 Java 平台选项

命名空间: **`aws:elasticbeanstalk:application:environment`**

名称	描述	默认值	有效值
JDBC_CONNECTION_STRING	外部数据库的连接字符串。	不适用	不适用

有关更多信息，请参阅 环境属性和其他软件设置。

命名空间: **`aws:elasticbeanstalk:container:tomcat:jvmoptions`**

名称	描述	默认值	有效值
JVM Options	在启动时将命令行选项传递给 JVM。	不适用	不适用
Xmx	最大 JVM 堆大小。	`256m`	不适用
XX:MaxPermSize	JVM 堆中用于存储类定义和相关元数据的部分。 **Note** 此选项仅适用于 Java 8 之前的 Java 版本，基于 Amazon Linux 2 和更高版本的 Elastic Beanstalk Tomcat 平台不支持此选项。	`64m`	不适用

名称	描述	默认值	有效值
Xms	初始 JVM 堆大小。	256m	不适用
optionName	除了 Tomcat 平台定义的选项外，还指定任意 JVM 选项。	不适用	不适用

命名空间: **aws:elasticbeanstalk:environment:proxy**

名称	描述	默认值	有效值
GzipCompression	设置为 false 以禁用响应压缩。 仅在 Amazon Linux AMI（在 Amazon Linux 2 之前）平台版本上有效。	true	true false
ProxyServer	设置要在环境的实例上使用的代理。如果将此选项设置为 apache，则 Elastic Beanstalk 将使用 Apache 2.4。 如果应用程序由于代理配置设置不兼容未准备好从 Apache 2.2 迁移，则设置为 apache/2.2 。此值仅在 Amazon Linux AMI（在 Amazon Linux 2 之前）平台版本上有效。 设置为 nginx 以使用 nginx。这是以 Amazon Linux 2 平台版本开始的默认值。 有关更多信息，请参阅配置 Tomcat 环境的代理服务器。	nginx (Amazon Linux 2) apache (Amazon Linux AMI)	apache apache/2.2 – 仅限 Amazon Linux AMI nginx

Linux 上的 .NET Core 平台选项

命名空间: **aws:elasticbeanstalk:environment:proxy**

名称	描述	默认值	有效值
ProxyServer	指定要用作代理的 Web 服务器。	nginx	nginx none

.NET 平台选项

命名空间: **aws:elasticbeanstalk:container:dotnet:apppool**

名称	描述	默认值	有效值
Target Runtime	选择用于应用程序的 .NET Framework 版本。	4.0	2.0 4.0
Enable 32-bit Applications	设置为 True 以运行 32 位应用程序。	False	True False

Node.js 平台选项

命名空间: **aws:elasticbeanstalk:environment:proxy**

名称	描述	默认值	有效值
ProxyServer	设置要在环境的实例上使用的代理。	nginx	apache nginx

Amazon Linux AMI（先前的 Amazon Linux 2）平台选项

命名空间: **`aws:elasticbeanstalk:container:nodejs`**

名称	描述	默认值	有效值
NodeCommand	用来启动 Node.js 应用程序的命令。 如果指定空字符串，则依次使用 `app.js`、`server.js` 和 `npm start`。	""	不适用
NodeVersion	Node.js 的版本。例如 `4.4.6` 支持的 Node.js 版本因 Node.js 平台版本而异。有关目前受支持版本的列表，请参阅 AWS Elastic Beanstalk 平台文档中的 Node.js。 **Note** 如果对您正在使用的 Node.js 版本的支持已从平台中移除，则您必须先更改或移除版本设置再进行平台更新。当在一个或多个 Node.js 版本中识别到安全漏洞时，可能会发生这种情况。 发生此情况时，尝试更新到不支持所配置的 NodeVersion 的新平台版本会失败。为避免需要创建新环境，请将 NodeVersion 配置选项更改为旧平台版本和新平台版本均支持的 Node.js 版本，或移除选项设置，然后执行平台更新。	变化	变化

GzipCompression	指定是否启用 gzip 压缩。如果将 ProxyServer 设置为 `none`，则会禁用 gzip 压缩。	`false`	`true`
			
			`false`

名称	描述	默认值	有效值
ProxyServer	指定应使用哪台 Web 服务器来代理与 Node.js 的连接。如果 ProxyServer 设置为 none，则静态文件映射不会生效，并且 gzip 压缩将被禁用。	nginx	apache nginx none

命名空间: **aws:elasticbeanstalk:container:nodejs:staticfiles**

您可以使用以下命名空间来配置代理服务器提供静态文件。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。这将减少您的应用程序必须处理的请求的数量。

在源代码中将代理服务器提供的路径映射到包含静态资产的文件夹。在此命名空间中定义的每个选项都映射不同的路径。

> **Note**
>
> 如果 aws:elasticbeanstalk:container:nodejs::ProxyFiles 设置为 none，将不应用静态文件设置。

名称	值
代理服务器将提供文件的路径。 示例：/images 将在 *subdomain* .elasticbeanstalk.com/images 提供文件。	包含文件的文件夹名称。 示例：staticimages 将在源代码包顶层从名为 staticimages 的文件夹中提供文件。

PHP 平台选项

命名空间: **`aws:elasticbeanstalk:container:php:phpini`**

名称	描述	默认值	有效值
document_root	指定项目的子目录，这也是面向公众的 Web 根目录。	`/`	将空字符串视为 `/`，或指定以 `/` 开头的字符串
memory_limit	分配给 PHP 环境的内存量。	`256M`	不适用

zlib.output_compression	指定 PHP 在输出时是否应使用压缩。	`Off`	`On`
			`Off`
			`true`
			`false`

allow_url_fopen	指定是否允许 PHP 的文件功能从远程位置 (如网站或 FTP 服务器) 检索数据。	`On`	`On`
			`Off`
			`true`
			`false`

display_errors	指定错误消息是否应该是输出的一部分。	`Off`	`On` `Off`
max_execution_time	设置一个脚本在被环境终止前允许运行的最长时间，计算单位为秒。	`60`	`0` 至 9223372036854775807 (PHP_INT_MAX)
composer_options	设置各种通过 composer.phar 安装并使用 Composer 安装依赖项时要使用的自定义选项。有关包括可用选项在内的更多信息，请转到 http://getcomposer.org/doc/03-cli.md#install。	不适用	不适用

命名空间: **aws:elasticbeanstalk:environment:proxy**

名称	描述	默认值	有效值
ProxyServer	设置要在环境的实例上使用的代理。	nginx	apache nginx

> **Note**
>
> 有关 PHP 平台的更多信息，请参阅使用 Elastic Beanstalk PHP 平台。

Python 平台选项

命名空间: **aws:elasticbeanstalk:application:environment**

名称	描述	默认值	有效值
DJANGO_SETTINGS_MODULE	指定要使用的设置文件。	不适用	不适用

有关更多信息，请参阅 环境属性和其他软件设置。

命名空间: **aws:elasticbeanstalk:container:python**

名称	描述	默认值	有效值
WSGIPath	包含 WSGI 应用程序的文件。此文件必须有一个可调用的 `application` 。	在 Amazon Linux 2 Python 平台版本上：`application` 在 Amazon Linux AMI Python	不适用

名称	描述	默认值	有效值
		平台版本上：application.py	
NumProcesses	运行 WSGI 应用程序时，应为进程组启动的守护程序进程数。	1	不适用
NumThreads	运行 WSGI 应用程序时，为了处理进程组内部每个守护程序进程中的请求而要创建的线程数。	15	不适用

命名空间: **aws:elasticbeanstalk:environment:proxy**

名称	描述	默认值	有效值
ProxyServer	设置要在环境的实例上使用的代理。	nginx	apache nginx

Amazon Linux AMI（先前的 Amazon Linux 2）平台选项

命名空间: **aws:elasticbeanstalk:container:python:staticfiles**

您可以使用以下命名空间来配置代理服务器提供静态文件。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。这将减少您的应用程序必须处理的请求的数量。

在源代码中将代理服务器提供的路径映射到包含静态资产的文件夹。在此命名空间中定义的每个选项都映射不同的路径。

默认情况下，Python 环境中的代理服务器提供 `static` 路径下名为 `/static` 的文件夹中的任何文件。

命名空间: **aws:elasticbeanstalk:container:python:staticfiles**

名称	值
代理服务器将提供文件的路径。	包含文件的文件夹名称。

名称	值
示例：/images 将在 *subdomain* .elasticbeanstalk.com/images 提供文件。	示例：staticimages 将在源代码包顶层从名为 staticimages 的文件夹中提供文件。

Ruby 平台选项

命名空间: **aws:elasticbeanstalk:application:environment**

名称	描述	默认值	有效值
RAILS_SKIP_MIGRATIONS	指定是代表用户的应用程序运行`rake db:migrate` ，还是应跳过它。此功能仅适用于 Rails 3 应用程序。	false	true false
RAILS_SKIP_ASSET_COMPILATION	指定该容器是应代表用户的应用程序运行`rake assets:precompile` ，还是应跳过它。此功能也仅适用于 Rails 3 应用程序。	false	true false
BUNDLE_WITHOUT	从 Gemfile 安装依赖项时要忽略的组列表，用冒号 (:) 分隔。	test:development	不适用
RACK_ENV	指定可以运行应用程序的环境阶段。常见的环境示例包括开发、生产和测试。	production	不适用

参阅 环境属性和其他软件设置 了解更多信息。

自定义选项

使用 aws:elasticbeanstalk:customoption 命名空间定义可在其他配置文件的 Resources 块中读取的选项和值。使用自定义选项收集用户在单个配置文件中指定的设置。

例如，您可能有一个复杂的配置文件，该文件定义了可由用户在启动环境时配置的资源。如果使用 Fn::GetOptionSetting 检索某个自定义选项的值，您可以在另一个配置文件中定义该选项，以便用户更容易发现和修改。

此外，由于自定义选项是配置选项，因此可在 API 级别设置它们以覆盖在配置文件中设置的值。有关更多信息，请参阅优先顺序。

自定义选项的定义方式与任何其他选项的类似：

```
option_settings:
  aws:elasticbeanstalk:customoption:
    option name: option value
```

例如，以下配置文件创建了一个名为 ELBAlarmEmail 的选项并将值设置为 someone@example.com：

```
option_settings:
  aws:elasticbeanstalk:customoption:
    ELBAlarmEmail: someone@example.com
```

在其他某个位置，配置文件定义了一个可读取包含 Fn::GetOptionSetting 的选项的 SNS 主题以填充 Endpoint 属性的值：

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint:
            Fn::GetOptionSetting:
              OptionName: ELBAlarmEmail
              DefaultValue: nobody@example.com
          Protocol: email
```

您可以在添加和自定义 Elastic Beanstalk 环境资源中找到更多使用 Fn::GetOptionSetting 的示例代码段。

使用配置文件 (.ebextensions) 进行高级环境自定义

您可以将 AWS Elastic Beanstalk 配置文件 (.ebextensions) 添加到 Web 应用程序的源代码中，以配置您的环境并自定义其包含的 AWS 资源。配置文件是带有.config文件扩展名的 YAML 或 JSON 格式的文档，您可以将其放在名为的文件夹中.ebextensions并部署在应用程序源包中。

Example .ebextensions/ .config network-load-balancer

此示例进行了简单的配置更改。它修改一个配置选项，以将环境的负载均衡器的类型设置为 Network Load Balancer。

```
option_settings:
  aws:elasticbeanstalk:environment:
    LoadBalancerType: network
```

我们建议对配置文件使用 YAML，因为它比 JSON 更可读。YAML 支持注释、多行命令、针对使用引号的多个备选项等。但是，您可以使用 YAML 或 JSON 在 Elastic Beanstalk 配置文件中以相同的方式进行任何配置更改。

> **提示**
>
> 当您开发或测试新的配置文件时，启动运行默认应用程序的干净环境并部署到此环境。格式不正确的配置文件将导致新环境启动失败，并且错误不可恢复。

配置文件的 `option_settings` 部分定义配置选项的值。配置选项允许您配置 Elastic Beanstalk 环境 AWS 、其中的资源以及运行应用程序的软件。设置配置选项有几种方法，配置文件只是其中之一。

该`Resources`部分允许您进一步自定义应用程序环境中的资源，并定义配置选项提供的功能之外的其他 AWS 资源。您可以添加和配置支持的任何资源 AWS CloudFormation，Elastic Beanstalk 使用这些资源来创建环境。

配置文件的其他部分
(`packages`、`sources`、`files`、`users`、`groups`、`commands`、`container_commands` 和 `services`) 可配置在您的环境中启动的 EC2 实例。只要您的环境中有服务器启动，Elastic Beanstalk 就会运行这些部分中定义的操作，为您的应用程序准备操作系统和存储系统。

有关常用 .ebextensions 的示例，请参阅 Elastic Beanstalk 配置文件存储库。

要求

- 位置 — Elastic Beanstalk 将`.ebextensions`处理您的部署中存在的所有文件夹。但是，我们建议您将所有配置文件放在源包根目录下名为`.ebextensions`的单个文件夹中。以点开头的文件夹可被文件浏览器隐藏，因此请确保在创建源包时添加文件夹。有关更多信息，请参阅 创建应用程序源包。。
- 命名 – 配置文件必须具有 `.config` 文件扩展名。
- 格式 – 配置文件必须符合 YAML 或 JSON 规范。

  当使用 YAML 时，请始终使用空格以不同的嵌套级别缩进键。有关 YAML 的更多信息，请参阅 YAML Ain't 标记语言 (YAML™) 版本 1.1。

- 唯一性 – 在每个配置文件中使用每个键一次。

> **Warning**
>
> 如果您在同一个配置文件中两次使用同一个键 (例如 `option_settings`)，则会删除两个部分之一。将重复的部分组合为单个部分，或将其放在不同的配置文件中。

根据您用来管理环境的客户端，部署过程可能略有不同。有关详细信息，请参阅下面几节：

- Elastic Beanstalk 控制台
- EB CLI
- AWS CLI

主题

选项设置

您可以使用 `option_settings` 键修改 Elastic Beanstalk 配置，并定义可以使用环境变量从应用程序中检索的变量。一些命名空间可让您扩展参数的数量，并指定参数名。有关命名空间和配置选项的列表，请参阅配置选项。

也可在创建或更新环境期间将选项设置直接应用于环境。直接应用于环境的设置将覆盖配置文件中相同选项的设置。移除环境配置中的设置后，配置文件中的设置将生效。有关更多信息，请参阅 优先顺序。

语法

选项设置标准语法是对象数组，每一个对象都有一个 `namespace`、`option_name` 和 `value` 键。

```
option_settings:
  - namespace:  namespace
    option_name:  option name
```

```
    value:  option value
  - namespace:  namespace
    option_name:  option name
    value:  option value
```

namespace 键可选。如果不指定命名空间，则默认使用 aws:elasticbeanstalk:application:environment：

```
option_settings:
  - option_name:  option name
    value:  option value
  - option_name:  option name
    value:  option value
```

Elastic Beanstalk 还支持选项设置快速输入语法，您可以在命名空间下以键值对形式指定选项：

```
option_settings:
  namespace:
    option name: option value
    option name: option value
```

示例

下面的示例在 aws:elasticbeanstalk:container:tomcat:jvmoptions 命名空间中设置一个特定于 Tomcat 平台的选项和一个名为 MYPARAMETER 的环境属性。

标准 YAML 格式：

Example .ebextensions/options.config

```
option_settings:
  - namespace:  aws:elasticbeanstalk:container:tomcat:jvmoptions
    option_name:  Xmx
    value:  256m
  - option_name: MYPARAMETER
    value: parametervalue
```

快速输入格式：

Example .ebextensions/options.config

```
option_settings:
```

```
  aws:elasticbeanstalk:container:tomcat:jvmoptions:
    Xmx: 256m
  aws:elasticbeanstalk:application:environment:
    MYPARAMETER: parametervalue
```

在 JSON 中：

Example .ebextensions/options.config

```
{
  "option_settings": [
    {
      "namespace": "aws:elasticbeanstalk:container:tomcat:jvmoptions",
      "option_name": "Xmx",
      "value": "256m"
    },
    {
      "option_name": "MYPARAMETER",
      "value": "parametervalue"
    }
  ]
}
```

自定义 Linux 服务器上的软件

建议自定义和配置您的应用程序所依赖的软件。您可以添加要在实例预置期间执行的命令；定义 Linux 用户和组；以及下载文件或直接在环境实例上创建文件。这些文件可能是应用程序所需的依赖项（例如 yum 存储库中的其他包），也可能是配置文件（例如代理配置的替代项，用于覆盖 Elastic Beanstalk 默认的特定设置）。

本节描述了您可以在配置文件中包含的信息类型，以便自定义运行 Linux 的 EC2 实例上的软件。有关自定义和配置 Elastic Beanstalk 环境的一般信息，请参阅配置 Elastic Beanstalk 环境。有关自定义运行 Windows 的 EC2 实例上的软件的信息，请参阅自定义 Windows Server 上的软件。

> **注意**
>
> - 在 Amazon Linux 2 平台上，我们强烈建议您使用 Buildfile，而不是在 .ebextensions 配置文件中提供文件和命令。Procfile 和平台挂钩，（如果可能）用于在实例预置期间在环境实例上配置和运行自定义代码。有关这些机制的详细信息，请参阅the section called “扩展 Linux 平台”。

- YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

配置文件支持以下键，它们影响运行您的应用程序的 Linux 服务器。

密钥

- 软件包
- 组
- 用户
- 源
- 文件
- 命令
- 服务
- 容器命令
- 示例：使用自定义 Amazon CloudWatch 指标

将按键在此处列出的顺序对其进行处理。

观察您的环境的事件，同时开发和测试配置文件。Elastic Beanstalk 将忽略包含验证错误的配置文件（例如“密钥无效”），并且不会处理同一文件中的任何其他密钥。出现这种情况时，Elastic Beanstalk 将警告事件添加到事件日志中。

软件包

您可以使用 packages 键下载并安装预先打包的应用程序和组件。

语法

```
packages:
  name of package manager:
    package name: version
    ...
  name of package manager:
    package name: version
    ...
  ...
```

您可以在每个程序包管理器密钥下指定多个软件包。

支持的软件包格式

目前，Elastic Beanstalk 支持以下包管理器：yum、rubygems、python 和 rpm。包的处理顺序如下：rpm、yum、rubygems 和 python。rubygems 和 python 之间没有处理顺序。在每个包管理器内，不保证包安装顺序。请使用您的操作系统支持的包管理器。

> **Note**
>
> Elastic Beanstalk 支持两个用于 Python 的基础包管理器：pip 和 easy_install。但是，在配置文件语法中，您必须将包管理器名称指定为 `python`。当您使用配置文件指定 Python 包管理器时，Elastic Beanstalk 会使用 Python 2.7。如果您的应用程序依赖于不同的 Python 版本，可以在 `requirements.txt` 文件中指定要安装的包。有关更多信息，请参阅使用要求文件指定依赖项。

指定版本

在每个包管理器内，会使用包名和一系列版本对每个包进行指定。版本可以是字符串、一系列版本、空字符串或者空列表。空字符串或者空列表表示您需要最新的版本。对于 rpm 管理器，版本是以磁盘或 URL 上的文件的路径的方式指定的。不支持相对路径。

如果您指定包的版本，Elastic Beanstalk 会尝试安装该版本，即使实例上已经安装了较新的包版本。如果已经安装了较新的版本，那么部署会失败。一些包管理器支持多个版本，但其他的包管理器可能不支持。有关更多信息，请查看相关的包管理器文档。如果您没有指定版本且已安装了某种版本的包，那么 Elastic Beanstalk 不会安装新的版本，因为它会认为您希望保留并使用现有的版本。

示例代码段

下面的代码段为 rpm 指定版本 URL，从 yum 请求最新版本并从 rubygems 请求 chef 的版本 0.10.2。

```
packages:
  yum:
    libmemcached: []
    ruby-devel: []
    gcc: []
  rpm:
    epel: http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm
  rubygems:
```

```
    chef: '0.10.2'
```

组

您可以使用 groups 键创建 Linux/UNIX 组并分配组 ID。要创建组，请添加新的密钥值对，将新的组名映射到可选组 ID。组键可以包含一个或多个组名。下表列出了可用的密钥。

语法

```
groups:
  name of group: {}
  name of group:
    gid: "group id"
```

选项

gid

组 ID 编号。

如果指定了组 ID，且存在该名称的组，那么创建组将失败。如果另一个组已拥有所指定的组 ID，那么操作系统可能会拒绝创建组。

示例代码段

以下代码段指定了一个名为“groupOne”、但没有分配组 ID 的组，以及一个名为“groupTwo”、但指定了组 ID 值为 45 的组。

```
groups:
  groupOne: {}
  groupTwo:
    gid: "45"
```

用户

您可以使用 users 键在 EC2 实例上创建 Linux/UNIX 用户。

语法

```
users:
```

```
  name of user:
    groups:
      - name of group
    uid: "id of the user"
    homeDir: "user's home directory"
```

选项

uid

用户 ID。如果用户名还有另一个用户 ID，那么此创建过程会失败。如果用户 ID 已指定给现有的用户，那么操作系统可能会拒绝该创建请求。

groups

一系列组名。用户会添加到列表中的每个组。

homeDir

用户的主目录。

创建的用户属于非交互式系统用户，带有 /sbin/nologin 外壳。这是特意设计的，无法修改。

示例代码段

```
users:
  myuser:
    groups:
      - group1
      - group2
    uid: "50"
    homeDir: "/tmp"
```

源

您可以使用 sources 密钥从公共 URL 下载存档文件并将其解压到 EC2 实例上的目标目录中。

语法

```
sources:
  target directory: location of archive file
```

支持的格式

支持的格式有 tar、tar+gzip、tar+bz2 和 Zip。只要该 URL 可公开访问，您就可以引用 Amazon Simple Storage Service (Amazon S3) 等外部位置 (例如，`https://mybucket.s3.amazonaws.com/myobject`)。

示例代码段

以下示例会从 Amazon S3 存储桶下载一个公有 .zip 文件并将其解压到 `/etc/myapp` 中：

```
sources:
  /etc/myapp: https://mybucket.s3.amazonaws.com/myobject
```

> **Note**
>
> 多个提取不应重用相同的目标路径。将其他源提取到项目的相同目标路径上将替换内容而不是附加到内容上。

文件

您可以使用 `files` 密钥在 EC2 实例上创建文件。该内容可以内嵌在配置文件中，也可以从 URL 中提取。这些文件会按词典顺序写入磁盘。

您可以提供一个用于授权的实例配置文件，从而使用 `files` 密钥从 Amazon S3 下载私有文件。

如果您指定的文件路径已在实例上存在，则将保留现有文件，并在其名称后附加扩展名 `.bak`。

语法

```
files:
  "target file location on disk":
     mode: "six-digit octal value"
     owner: name of owning user for file
     group: name of owning group for file
     source: URL
     authentication: authentication name:

  "target file location on disk":
     mode: "six-digit octal value"
     owner: name of owning user for file
```

```
    group: name of owning group for file
    content: |
     # this is my
     # file content
    encoding: encoding format
    authentication: authentication name:
```

选项

`content`

要添加到文件的字符串内容。指定 `content` 或 `source`，但不能同时指定两者。

`source`

要下载的文件的 URL。指定 `content` 或 `source`，但不能同时指定两者。

`encoding`

使用 `content` 选项指定的字符串的编码格式。

有效值：`plain` | `base64`

`group`

拥有文件的 Linux 组。

`owner`

拥有文件的 Linux 用户。

`mode`

是一个六位数的八进制值，表示这个文件的模式。在 Windows 系统中不受支持。将前三位数用于符号链接，将后三位数用于设置权限。要创建符号链接，请指定 `120`*`xxx`*，其中 `xxx` 定义目标文件的权限。要指定文件的权限，请使用最后三位，例如 `000644`。

`authentication`

要使用的 AWS CloudFormation 身份验证方法的名称。您可以使用资源键将身份验证方法添加到 Auto Scaling 组元数据。有关示例，请参阅以下内容。

示例代码段

```
files:
```

```
  "/home/ec2-user/myfile" :
    mode: "000755"
    owner: root
    group: root
    source: http://foo.bar/myfile

  "/home/ec2-user/myfile2" :
    mode: "000755"
    owner: root
    group: root
    content: |
      this is my
      file content
```

使用符号链接的示例。这会创建指向现有文件 /tmp/myfile2.txt 的链接 /tmp/myfile1.txt。

```
files:
  "/tmp/myfile2.txt" :
    mode: "120400"
    content: "/tmp/myfile1.txt"
```

以下示例使用资源键添加名为 S3Auth 的身份验证方法并使用该方法从 Amazon S3 存储桶下载私有文件：

```
Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-west-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"

files:
  "/tmp/data.json" :
    mode: "000755"
    owner: root
    group: root
    authentication: "S3Auth"
```

```
    source: https://elasticbeanstalk-us-west-2-123456789012.s3-us-west-2.amazonaws.com/
data.json
```

命令

您可以使用 commands 键在 EC2 实例上执行命令。这些命令在应用程序之前运行，设置 Web 服务器以及提取应用程序版本文件。

指定的命令以根用户身份运行且按名称的字母顺序进行处理。默认情况下，该命令在根目录中运行。要从另一个目录运行命令，请使用 cwd 选项。

要使用您的命令排查问题，您可以在实例日志中查找其输出。

语法

```
commands:
  command name:
    command: command to run
    cwd: working directory
    env:
      variable name: variable value
    test: conditions for command
    ignoreErrors: true
```

选项

command

一个数组（YAML 语法中的数据块序列集合）或一个用于指定要运行的命令的字符串。一些重要说明：

- 如果您使用了字符串，则不需要用引号将整个字符串引起来。如果您确实使用了引号，请对相同类型的引号的文本匹配项进行转义。
- 如果您使用了数组，则不需要对空格字符进行转义或用引号将命令参数引起来。每个数组元素都是一个命令参数。请勿使用数组指定多个命令。

以下示例具有完全相同的效果：

```
commands:
  command1:
    command: git commit -m "This is a comment."
  command2:
```

```
    command: "git commit -m \"This is a comment.\""
  command3:
    command: 'git commit -m "This is a comment."'
  command4:
    command:
      - git
      - commit
      - -m
      - This is a comment.
```

要指定多个命令，请使用文本数据块标量，如以下示例所示。

```
commands:
  command block:
    command: |
      git commit -m "This is a comment."
      git push
```

env

(可选) 设置该命令的环境变量。这个属性会覆盖而不是追加到现有的环境。

cwd

(可选) 工作目录。如果未指定，命令将从根目录 (/) 运行。

test

（可选）一个命令，此命令必须返回值 `true`（退出代码为 0），以便 Elastic Beanstalk 处理 `command` 键中包含的命令，例如 shell 脚本。

ignoreErrors

(可选) 一个布尔值，该值确定在 `command` 密钥中包含的命令失败时 (返回非零值) 是否应该运行其他命令。如果要在命令失败后继续运行其他命令，请将该值设置为 `true`。否则，请将该值设置为 `false`。默认值为 `false`。

示例代码段

以下示例代码段运行 Python 脚本。

```
commands:
  python_install:
    command: myscript.py
```

```
    cwd: /home/ec2-user
    env:
      myvarname: myvarvalue
    test: "[ -x /usr/bin/python ]"
```

服务

您可以使用 services 键定义哪些服务应该在实例启动时启动或者停止。services 键还可让您在源、包和文件上指定依赖项，以便在由于文件安装而需要重启时，让 Elastic Beanstalk 负责服务的重启。

语法

```
services:
  sysvinit:
    name of service:
      enabled: "true"
      ensureRunning: "true"
      files:
        - "file name"
      sources:
        - "directory"
      packages:
        name of package manager:
          "package name[: version]"
      commands:
        - "name of command"
```

选项

`ensureRunning`

设置为 `true` 可确保在 Elastic Beanstalk 完成后服务能够运行。

设置为 `false` 可确保该服务在 Elastic Beanstalk 完成后不运行。

忽略该密钥，将不更改服务状态。

`enabled`

设置为 `true`，可确保该服务在启动时自动启动。

设置为 `false`，可确保该服务在启动时不会自动启动。

忽略该密钥，将不更改这个属性。

`files`

一系列文件。如果 Elastic Beanstalk 直接通过文件块更改一个文件，则该服务会重启。

`sources`

一系列目录。如果 Elastic Beanstalk 将存档展开到这些目录中的一个，则该服务会重启。

`packages`

包管理器与一系列包名的映射。如果 Elastic Beanstalk 安装或者更新了这些包中的一个，则该服务会重启。

`commands`

一系列命令名称。如果 Elastic Beanstalk 运行指定的命令，则服务将重新启动。

示例代码段

以下是示例代码段：

```
services:
  sysvinit:
    myservice:
      enabled: true
      ensureRunning: true
```

容器命令

您可以使用 `container_commands` 键执行将影响应用程序源代码的命令。容器命令将在设置应用程序和 Web 服务器并提取应用程序版本存档后，但在部署应用程序版本前运行。非容器命令和其他自定义操作将在提取应用程序源代码之前执行。

指定的命令以根用户身份运行且按名称的字母顺序进行处理。容器命令从暂存目录运行，您的源代码在部署到应用程序服务器前在这里提取。当源部署到其最终位置时，将包含您在暂存目录中使用容器命令对源代码所做的任何更改。

> **Note**
>
> 您的容器命令的输出记录在 `cfn-init-cmd.log` 实例日志中。有关检索和查看实例日志的更多信息，请参阅查看 Amazon EC2 实例的日志。

可以使用 leader_only 仅对单个实例运行此命令，也可以配置 test 以仅在测试命令的评估结果为 true 时运行此命令。仅领导容器命令只在创建和部署环境期间执行，而其他命令和服务器自定义操作将在预配置或更新实例时执行。由于启动配置发生更改 (例如 AMI Id 或实例类型的更改)，未执行仅领导容器命令。

语法

```
container_commands:
  name of container_command:
    command: "command to run"
    leader_only: true
  name of container_command:
    command: "command to run"
```

选项

command

要运行的字符串或字符串数组。

env

(可选) 在运行命令之前设置环境变量，并覆盖任何现有值。

cwd

(可选) 工作目录。默认情况下，这是解压缩应用程序的暂存目录。

leader_only

（可选）仅对 Elastic Beanstalk 选定的单个实例运行命令。仅领导容器命令将在其他容器命令之前运行。命令可以是仅领导命令或具有 test，但不能同时具备这两个特性 (leader_only 优先)。

test

(可选) 运行必须返回 true 的测试命令以便运行容器命令。命令可以是仅领导命令或具有 test，但不能同时具备这两个特性 (leader_only 优先)。

ignoreErrors

(可选) 不在容器命令返回 0 以外的值 (成功) 时导致部署失败。设置为 true 可实现这一点。

示例代码段

以下是示例代码段。

```
container_commands:
  collectstatic:
    command: "django-admin.py collectstatic --noinput"
  01syncdb:
    command: "django-admin.py syncdb --noinput"
    leader_only: true
  02migrate:
    command: "django-admin.py migrate"
    leader_only: true
  99customize:
    command: "scripts/customize.sh"
```

示例：使用自定义 Amazon CloudWatch 指标

Amazon CloudWatch 是一项网络服务，可让您监控、管理和发布各种指标，并根据指标中的数据配置警报操作。您可以定义自定义指标供自己使用，Elastic Beanstalk 会将这些指标推送到亚马逊。CloudWatch一旦亚马逊 CloudWatch 包含您的自定义指标，您就可以在亚马逊 CloudWatch 控制台中查看这些指标。

> **Important**
>
> Amazon CloudWatch 监控脚本已被弃用。 CloudWatch 代理现在已经取代了用于收集指标和日志的 CloudWatch监控脚本。
> 如果您仍在从已弃用的监控脚本迁移到代理，并且需要有关监控脚本的信息，请参阅 Amazon EC2 用户指南中的已弃用：使用 CloudWatch 监控脚本收集指标。

亚马逊 CloudWatch 代理

Amazon CloudWatch 代理支持跨操作系统从 Amazon EC2 实例和本地服务器收集 CloudWatch 指标和日志。代理支持在系统级别收集的指标。此外，它还支持从应用程序或服务中收集自定义日志和指标。有关亚马逊 CloudWatch 代理的更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的通过 CloudWatch 代理收集指标和日志。

> **Note**
>
> Ela stic Beanstalk 增强型运行状况报告原生支持将各种实例和环境指标发布到。 CloudWatch 有关详细信息，请参阅 发布环境的 Amazon CloudWatch 自定义指标。

主题

.Ebextensions 配置文件

此示例使用.ebextensions 配置文件中的文件和命令在 Amazon Linux 2 平台 CloudWatch 上配置和运行亚马逊代理。此代理预先包装在 Amazon Linux 2 中。如果您使用的是其他操作系统，则可能需要执行其他安装代理的步骤。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的安装 CloudWatch 代理。

要使用该示例，请将其保存到名为 `cloudwatch.config` 的文件中，并将该文件放到项目目录顶层的 `.ebextensions` 目录中，然后使用 Elastic Beanstalk 控制台（在源包中包括 .ebextensions 目录）或 EB CLI 部署您的应用程序。

有关配置文件的详细信息，请参阅 使用配置文件 (.ebextensions) 进行高级环境自定义。

.ebextensions/cloudwatch.config

```
files:
  "/opt/aws/amazon-cloudwatch-agent/bin/config.json":
    mode: "000600"
    owner: root
    group: root
    content: |
      {
        "agent": {
          "metrics_collection_interval": 60,
          "run_as_user": "root"
        },
        "metrics": {
          "namespace": "System/Linux",
          "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
          },
          "metrics_collected": {
            "mem": {
              "measurement": [
                "mem_used_percent"
```

```
              ]
            }
          }
        }
      }
container_commands:
  start_cloudwatch_agent:
    command: /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
```

此文件有以下两个部分：

- `files` — 此部分添加代理配置文件。它指示代理应向 Amazon 发送哪些指标和日志 CloudWatch。在此示例中，我们只发送 mem_used_percent 指标。有关亚马逊 CloudWatch 代理支持的系统级指标的完整列表，请参阅《亚马逊 CloudWatch 用户指南》中 CloudWatch 代理收集的指标。
- `container_commands` — 本部分包含启动代理的命令，并将配置文件作为参数传递。有关 `container_commands` 的更多详细信息，请参阅 容器命令。

权限

您环境中的实例需要适当的 IAM 权限才能使用亚马逊 CloudWatch 代理发布自定义亚马逊 CloudWatch 指标。您可以将您的环境中的实例添加到此环境的实例配置文件中，从而向它们授予权限。您可以在部署应用程序前/后为此实例配置文件添加权限。

授予发布 CloudWatch 指标的权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 选择您的环境的实例配置文件角色。在使用 Elastic Beanstalk 控制台或 EB CLI 创建环境时，该角色默认为 `aws-elasticbeanstalk-ec2-role`。
4. 选择权限选项卡。
5. 在 Permissions Policies（权限策略）下，Permissions（权限）部分中，选择 Attach policies（附加策略）。
6. 在 “附加权限” 下，选择 AWS 托管策略CloudWatchAgentServerPolicy。然后单击 Attach policy（附加策略）。

有关如何管理策略的更多信息，请参阅《IAM 用户指南》中的使用策略。

在 CloudWatch 控制台中查看指标

将 CloudWatch 配置文件部署到您的环境后，请查看 Amazon CloudWatch 控制台以查看您的指标。自定义指标将位于 CWAgent 命名空间。

有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的查看可用指标。

自定义 Windows Server 上的软件

建议自定义和配置您的应用程序所依赖的软件。这些文件可能是该应用程序所要求的依赖项，例如需要运行的其他软件包或服务。有关自定义和配置 Elastic Beanstalk 环境的一般信息，请参阅配置 Elastic Beanstalk 环境。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

配置文件支持以下可对运行您的应用程序的 Windows 服务器产生影响的键。

密钥

- 软件包
- 源
- 文件
- 命令
- 服务
- 容器命令

将按键在此处列出的顺序对其进行处理。

> **Note**
>
> 较旧（不受版本控制）的 .NET 平台版本不会按正确顺序处理配置文件。有关更多信息，请参阅跨 Elastic BeanstalkWindows Server 平台的主要版本迁移。

观察您的环境的事件，同时开发和测试配置文件。Elastic Beanstalk 将忽略包含验证错误的配置文件（例如“密钥无效”），并且不会处理同一文件中的任何其他密钥。出现这种情况时，Elastic Beanstalk 将警告事件添加到事件日志中。

软件包

使用 `packages` 键下载并安装预先打包的应用程序和组件。

在 Windows 环境中，Elastic Beanstalk 支持下载和安装 MSI 软件包。（Linux 环境支持其他软件包管理器。有关详细信息，请参阅在 Linux 服务器上自定义软件页面上的 软件包。）

只要该 URL 可公开访问，您就可以引用 Amazon Simple Storage Service (Amazon S3) 对象等任意外部位置。

如果您指定了多个 `msi:` 程序包，则无法保证其安装顺序。

语法

指定您选择的名称作为软件包名称，而且指定 MSI 文件位置的 URL 作为该值。您可以在 `msi:` 密钥下指定多个程序包。

```
packages:
  msi:
    package name: package url
    ...
```

示例

下面的示例指定了从 `https://dev.mysql.com/` 中下载 mysql 的 URL。

```
packages:
  msi:
    mysql: https://dev.mysql.com/get/Downloads/Connector-Net/mysql-connector-
net-8.0.11.msi
```

以下示例指定了 Amazon S3 对象作为 MSI 文件位置。

```
packages:
  msi:
    mymsi: https://mybucket.s3.amazonaws.com/myobject.msi
```

源

使用 `sources` 键从公有 URL 下载存档文件并将其解压到 EC2 实例上的目标目录中。

语法

```
sources:
  target directory: location of archive file
```

支持的格式

在 Windows 环境中，Elastic Beanstalk 支持 .zip 格式。（Linux 环境支持其他格式。有关详细信息，请参阅在 Linux 服务器上自定义软件页面上的 源。）

只要该 URL 可公开访问，您就可以引用 Amazon Simple Storage Service (Amazon S3) 对象等任意外部位置。

示例

以下示例会从 Amazon S3 存储桶下载一个公有 .zip 文件并将其解压到 `c:/myproject/myapp` 中。

```
sources:
  "c:/myproject/myapp": https://mybucket.s3.amazonaws.com/myobject.zip
```

文件

使用 `files` 键在 EC2 实例上创建文件。内容可以内嵌在配置文件中，也可以取自 URL。这些文件会按词典顺序写入磁盘。要从 Amazon S3 下载私有文件，请提供实例配置文件以进行授权。

语法

```
files:
  "target file location on disk":
    source: URL
    authentication: authentication name:

  "target file location on disk":
    content: |
      this is my content
    encoding: encoding format
```

选项

`content`

(可选) 一个字符串。

`source`

(可选) 加载文件的 URL。不能使用内容密钥指定该选项。

`encoding`

(可选) 编码格式。此选项只用于所提供的内容键值。默认值为 `plain`。

有效值：`plain` | `base64`

`authentication`

（可选）要使用的 AWS CloudFormation 身份验证方法的名称。您可以使用资源键将身份验证方法添加到 Auto Scaling 组元数据。

示例

以下示例显示了提供文件内容的两种方式：通过 URL 或内嵌在配置文件中。

```
files:
  "c:\\targetdirectory\\targetfile.txt":
    source: http://foo.bar/myfile

  "c:/targetdirectory/targetfile.txt":
    content: |
      # this is my file
      # with content
```

> **Note**
>
> 如果您在文件路径中使用反斜杠 (\)，则必须在它前面再插入一个反斜杠 (转义字符)，如上例所示。

以下示例使用资源键添加名为 S3Auth 的身份验证方法并使用该方法从 Amazon S3 存储桶下载私有文件：

```
files:
  "c:\\targetdirectory\\targetfile.zip":
    source: https://elasticbeanstalk-us-east-2-123456789012.s3.amazonaws.com/prefix/
myfile.zip
    authentication: S3Auth

Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-east-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"
```

命令

使用 commands 键在 EC2 实例上执行命令。该命令是按名称的字母顺序处理的，它们会在安装应用程序和 Web 服务器以及提取应用程序版本文件前运行。

管理员用户身份可运行指定的命令。

要使用您的命令排查问题，您可以在实例日志中查找其输出。

语法

```
commands:
  command name:
    command: command to run
```

选项

command

指定要运行的命令的数组或字符串。如果您使用了数组，则不需要对空格字符进行转义或用引号将命令参数引起来。

`cwd`

(可选) 工作目录。默认情况下，Elastic Beanstalk 会尝试查找项目的目录位置。如果找不到，则它会使用 `c:\Windows\System32` 作为默认目录位置。

`env`

(可选) 设置该命令的环境变量。这个属性会覆盖而不是追加到现有的环境。

`ignoreErrors`

（可选）一个布尔值，该值确定在 `command` 密钥中包含的命令失败时（返回非零值）是否应该运行其他命令。如果要在命令失败后继续运行其他命令，请将该值设置为 `true`。否则，请将该值设置为 `false`。默认值为 `false`。

`test`

（可选）一种命令，该命令必须返回值 `true`（退出代码 0），以便 Elastic Beanstalk 处理 `command` 键中包含的命令。

`waitAfterCompletion`

(可选) 在命令完成后、运行下一个命令前等待的秒数。如果系统需要在命令完成后重启，则会在经过指定的秒数后重启系统。如果系统因某个命令而重启，Elastic Beanstalk 将恢复到配置文件中该命令之后的点。默认值为 **`60`** 秒。您还可以指定 **`forever`**，但须重启系统后才能运行其他命令。

示例

下面的示例将 `set` 命令的输出保存到指定的文件。如果有后续命令，Elastic Beanstalk 会在此命令完成后立即运行后续命令。如果命令需要重新启动，则 Elastic Beanstalk 会在命令完成后立即重新启动实例。

```
commands:
  test:
    command: set > c:\\myapp\\set.txt
    waitAfterCompletion: 0
```

服务

使用 `services` 键定义哪些服务应该在实例启动时启动或者停止。`services` 键还可让您指定源、包和文件的依赖项，以便在由于文件安装而需要重启时，让 Elastic Beanstalk 负责服务的重启。

语法

```
services:
  windows:
    name of service:
      files:
        - "file name"
      sources:
        - "directory"
      packages:
        name of package manager:
            "package name[: version]"
      commands:
        - "name of command"
```

选项

`ensureRunning`

（可选）设置为 `true` 可确保在 Elastic Beanstalk 完成后服务能够运行。

设置为 `false` 可确保该服务在 Elastic Beanstalk 完成后不运行。

忽略该密钥，将不更改服务状态。

`enabled`

(可选) 设为 `true` 以确保此服务在启动时自动启动。

设置为 `false`，可确保该服务在启动时不会自动启动。

忽略该密钥，将不更改这个属性。

`files`

一系列文件。如果 Elastic Beanstalk 直接通过文件块更改一个文件，则该服务会重启。

`sources`

一系列目录。如果 Elastic Beanstalk 将存档展开到这些目录中的一个，则该服务会重启。

`packages`

包管理器与一系列包名的映射。如果 Elastic Beanstalk 安装或者更新了这些包中的一个，则该服务会重启。

commands

一系列命令名称。如果 Elastic Beanstalk 运行指定的命令，则服务将重新启动。

示例

```
services:
  windows:
    myservice:
      enabled: true
      ensureRunning: true
```

容器命令

使用 container_commands 键执行将影响应用程序源代码的命令。容器命令将在设置应用程序和 Web 服务器并提取应用程序版本存档后，但在部署应用程序版本前运行。非容器命令和其他自定义操作将在提取应用程序源代码之前执行。

容器命令从暂存目录运行，您的源代码在部署到应用程序服务器前在这里提取。当源部署到其最终位置时，将包含您在暂存目录中使用容器命令对源代码所做的任何更改。

要使用您的容器命令排查问题，您可以在实例日志中查找其输出。

可以使用 leader_only 选项只对单个实例运行此命令，也可以配置 test 以只在测试命令的评估结果为 true 时运行此命令。仅领导容器命令只在创建和部署环境期间执行，而其他命令和服务器自定义操作将在预配置或更新实例时执行。由于启动配置发生更改 (例如 AMI Id 或实例类型的更改)，未执行仅领导容器命令。

语法

```
container_commands:
  name of container_command:
    command: command to run
```

选项

command

要运行的字符串或字符串数组。

`env`

(可选) 在运行命令之前设置环境变量，并覆盖任何现有值。

`cwd`

(可选) 工作目录。默认情况下，这是解压缩应用程序的暂存目录。

`leader_only`

（可选）仅对 Elastic Beanstalk 选定的单个实例运行命令。仅领导容器命令将在其他容器命令之前运行。命令可以是仅领导命令或具有 `test`，但不能同时具备这两个特性 (`leader_only` 优先)。

`test`

(可选) 运行必须返回 `true` 的测试命令以便运行容器命令。命令可以是仅领导命令或具有 `test`，但不能同时具备这两个特性 (`leader_only` 优先)。

`ignoreErrors`

(可选) 不在容器命令返回 0 以外的值 (成功) 时导致部署失败。设置为 `true` 可实现这一点。

`waitAfterCompletion`

(可选) 在命令完成后、运行下一个命令前等待的秒数。如果系统需要在命令完成后重启，则会在经过指定的秒数后重启系统。如果系统因某个命令而重启，Elastic Beanstalk 将恢复到配置文件中该命令之后的点。默认值为 **60** 秒。您还可以指定 **forever**，但须重启系统后才能运行其他命令。

示例

下面的示例将 `set` 命令的输出保存到指定的文件。Elastic Beanstalk 在一个实例上运行此命令，并在命令完成后立即重新启动实例。

```
container_commands:
  foo:
    command: set > c:\\myapp\\set.txt
    leader_only: true
    waitAfterCompletion: 0
```

添加和自定义 Elastic Beanstalk 环境资源

您可能希望自定义属于 Elastic Beanstalk 环境一部分的环境资源。例如，您可能会希望添加 Amazon SQS 队列和有关队列深度的警报，或者添加 Amazon ElastiCache 集群。在部署应用程序版本的同时，您就可以轻松地自定义环境，只需使用源包将配置文件添加进来即可。

您可以使用配置文件中的 `Resources` 键在环境中创建和自定义 AWS 资源。在配置文件中定义的资源会添加到用于启动环境的 AWS CloudFormation 模板中。所有 AWS CloudFormation 资源类型均受支持。

> **Note**
>
> 无论何时添加不由 Elastic Beanstalk 管理的资源，请务必向 AWS Identity and Access Management (IAM) 用户添加具有适当权限的用户策略。Elastic Beanstalk 提供的托管用户策略仅涵盖对 Elastic Beanstalk 托管资源的权限。

例如，以下配置文件将 Auto Scaling 生命周期挂钩添加到 Elastic Beanstalk 创建的默认 Auto Scaling 组中：

~/my-app/.ebextensions/as-hook.config

```
Resources:
  hookrole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument: {
               "Version" : "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "autoscaling.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            }
      Policies: [ {
               "PolicyName": "SNS",
               "PolicyDocument": {
                      "Version": "2012-10-17",
                      "Statement": [{
                          "Effect": "Allow",
                          "Resource": "*",
                          "Action": [
                              "sqs:SendMessage",
                              "sqs:GetQueueUrl",
                              "sns:Publish"
                          ]
```

```
                    }
                  ]
               }
            } ]
  hooktopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: "my-email@example.com"
          Protocol: email
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      AutoScalingGroupName: { "Ref" : "AWSEBAutoScalingGroup" }
      LifecycleTransition: autoscaling:EC2_INSTANCE_TERMINATING
      NotificationTargetARN: { "Ref" : "hooktopic" }
      RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }
```

此示例定义三个资源：hookrole、hooktopic 和 lifecyclehook。前两个资源分别是 IAM 角色和 SNS 主题，前者授予 Amazon EC2 Auto Scaling 向 Amazon SNS 发布消息的权限，后者将来自 Auto Scaling 组的消息中继到电子邮件地址。Elastic Beanstalk 使用指定的属性和类型创建这些资源。

最后一个资源 lifecyclehook 是生命周期钩子本身：

```
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      AutoScalingGroupName: { "Ref" : "AWSEBAutoScalingGroup" }
      LifecycleTransition: autoscaling:EC2_INSTANCE_TERMINATING
      NotificationTargetARN: { "Ref" : "hooktopic" }
      RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }
```

生命周期钩子使用两个函数填充钩子属性值。{ "Ref" : "AWSEBAutoScalingGroup" } 检索 Elastic Beanstalk 为环境创建的 Auto Scaling 组的名称。AWSEBAutoScalingGroup 是 Elastic Beanstalk 提供的标准资源名称之一。

对于 AWS::IAM::Role，Ref 只返回角色名称，而不返回 ARN。要获取 RoleARN 参数的 ARN，需要改用另一个内部函数 Fn::GetAtt，该函数获取一个资源的所有属性。RoleARN: { "Fn::GetAtt" : ["hookrole", "Arn"] } 将从 Arn 资源获取 hookrole 属性。

`{ "Ref" : "hooktopic" }` 获取之前在配置文件中创建的 Amazon SNS 主题的 ARN。`Ref` 返回的值因资源类型而异，可以在 AWS CloudFormation 用户指南的 AWS::SNS::Topic 资源类型主题中找到。

修改 Elastic Beanstalk 为环境创建的资源

Elastic Beanstalk 为您的环境创建的资源有名称。您可以使用这些名称通过函数来获取有关资源的信息，或修改资源中的属性来自定义其行为。本主题介绍 Elastic Beanstalk 在不同类型的环境中使用的 AWS 资源。

> **Note**
>
> 上一主题自定义资源提供了一些用于自定义环境资源的使用案例和示例。另外，您可以在后面的主题 自定义资源示例 中找到更多配置文件示例。

Web 服务器环境具有以下资源。

Web 服务器环境

- `AWSEBAutoScalingGroup` (AWS::AutoScaling::AutoScalingGroup) – 附加到环境的 Auto Scaling 组。
- 以下两种资源之一。
  - `AWSEBAutoScalingLaunchConfiguration` (AWS::AutoScaling::LaunchConfiguration) – 附加到环境的 Auto Scaling 组中的启动配置。
  - `AWSEBEC2LaunchTemplate` (AWS::EC2::LaunchTemplate) – 环境的 Auto Scaling 组使用的 Amazon EC2 启动模板。

  > **Note**
  >
  > 如果您的环境使用需要 Amazon EC2 启动模板的功能，并且您的用户策略缺少所需权限，则创建或更新环境可能会失败。使用 AdministratorAccess-AWSElasticBeanstalk 托管用户策略，或者将所需权限添加到您的自定义策略。
- `AWSEBEnvironmentName` (AWS::ElasticBeanstalk::Environment) – 您的环境。
- `AWSEBSecurityGroup` (AWS::EC2::SecurityGroup) – 附加到 Auto Scaling 组的安全组。
- `AWSEBRDSDatabase` (AWS::RDS::DBInstance) – 附加到您的环境的 Amazon RDS 数据库实例（如果适用）。

在负载均衡的环境中，您可以访问与负载均衡器相关的其他资源。传统负载均衡器有一个资源用于负载均衡器，一个资源用于连接到它的安全组。应用程序和网络负载均衡器还有附加资源可用于负载均衡器的默认侦听器、侦听器规则和目标组。

负载均衡环境

- `AWSEBLoadBalancer` (AWS::ElasticLoadBalancing::LoadBalancer) – 您的环境的 Classic Load Balancer。
- `AWSEBV2LoadBalancer` (AWS::ElasticLoadBalancingV2::LoadBalancer) – 您的环境的应用程序或 Network Load Balancer。
- `AWSEBLoadBalancerSecurityGroup` (AWS::EC2::SecurityGroup) – 仅在自定义 Amazon Virtual Private Cloud (Amazon VPC) 中，Elastic Beanstalk 为负载均衡器创建的安全组的名称。在默认 VPC 或 EC2 Classic 中，Elastic Load Balancing 会将默认安全组分配给负载均衡器。
- `AWSEBV2LoadBalancerListener` (AWS::ElasticLoadBalancingV2::Listener) – 一个侦听器，它允许负载均衡器检查连接请求并将其转发给一个或多个目标组。
- `AWSEBV2LoadBalancerListenerRule` (AWS::ElasticLoadBalancingV2::ListenerRule) – 定义 Elastic Load Balancing 侦听器对哪些请求执行操作以及所执行的操作。
- `AWSEBV2LoadBalancerTargetGroup` (AWS::ElasticLoadBalancingV2::TargetGroup) – 一个 Elastic Load Balancing 目标组，将请求路由到一个或多个注册目标，例如 Amazon EC2 实例。

工作线程环境具有用于可缓存传入请求的 SQS 队列的资源，还有一个 Amazon DynamoDB 表，实例可使用该表来选择领导。

工作线程环境

- `AWSEBWorkerQueue` (AWS::SQS::Queue) – 守护程序从中拉取所需处理请求的 Amazon SQS 队列。
- `AWSEBWorkerDeadLetterQueue` (AWS::SQS::Queue) – 存储无法送达或守护程序未成功处理的消息的 Amazon SQS 队列。
- `AWSEBWorkerCronLeaderRegistry` (AWS::DynamoDB::Table) – Amazon DynamoDB 表，它是守护进程用于周期任务的内部注册表。

其他 AWS CloudFormation 模板密钥

我们已经引入了 AWS CloudFormation 诸如`Resources`、`files`、和之类的配置文件密钥`packages`。Elastic Beanstalk 将配置文件 AWS CloudFormation 内容添加到支持您的环境的模板中，因此 AWS CloudFormation 您可以使用其他部分在配置文件中执行高级任务。

键

- 参数
- 输出
- 映像

参数

参数是 Elastic Beanstalk 自己的自定义选项的替代项，您可以用来定义要在配置文件的其他位置使用的值。与自定义选项相似，您可以使用参数在一个位置，收集用户可配置值。与自定义选项不同，您不能使用 Elastic Beanstalk 的 API 来设置参数值，并且可以在模板中定义的参数数量受到限制。 AWS CloudFormation

您可能想要使用参数的原因之一是让您的配置文件兼作 AWS CloudFormation 模板。如果您使用参数而不是自定义选项，则可以使用配置文件在中 AWS CloudFormation 创建与其自己的堆栈相同的资源。例如，您可能有一个配置文件用于向环境中添加用于测试的 Amazon EFS 文件系统，然后使用同一个文件创建独立的文件系统，不绑定到用于生产用途的环境生命周期。

以下示例演示使用参数在配置文件顶部收集用户可配置的值。

Example L oadbalancer-accesslogs-existingbucket .config — 参数

```
Parameters:
  bucket:
    Type: String
    Description: "Name of the Amazon S3 bucket in which to store load balancer logs"
    Default: "DOC-EXAMPLE-BUCKET"
  bucketprefix:
    Type: String
    Description: "Optional prefix. Can't start or end with a /, or contain the word
 AWSLogs"
    Default: ""
```

输出

您可以使用 Outputs 数据块，将有关已创建资源的信息导出到 AWS CloudFormation。然后，你可以使用该Fn::ImportValue函数将该值提取到 Elastic Beanstalk 之外的 AWS CloudFormation 模板中。

以下示例创建了一个 Amazon SNS 主题，并将其 ARN 导出到名称为。 AWS CloudFormation NotificationTopicArn

Example sns-topic.config

```
Resources:
  NotificationTopic:
    Type: AWS::SNS::Topic

Outputs:
  NotificationTopicArn:
    Description: Notification topic ARN
    Value: { "Ref" : "NotificationTopic" }
    Export:
      Name: NotificationTopicArn
```

在其他环境的配置文件或 Elastic Beanstalk 之外的 AWS CloudFormation 模板中，Fn::ImportValue您可以使用该函数来获取导出的 ARN。此示例将导出的值分配给名为 TOPIC_ARN 的环境属性。

Example env.config

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    TOPIC_ARN: '`{ "Fn::ImportValue" : "NotificationTopicArn" }`'
```

映像

您可以使用映射，来存储按命名空间组织的键-值对。映射可以帮助组织您在配置中使用的值，或根据其它值来更改参数值。例如，以下配置根据当前区域来设置账户 ID 参数的值。

Example L oadbalancer-accesslogs-newbucket .config — 映射

```
Mappings:
```

```
  Region2ELBAccountId:
    us-east-1:
      AccountId: "111122223333"
    us-west-2:
      AccountId: "444455556666"
    us-west-1:
      AccountId: "123456789012"
    eu-west-1:
      AccountId: "777788889999"
...
            Principal:
              AWS:
                ? "Fn::FindInMap"
                :
                  - Region2ELBAccountId
                  -
                    Ref: "AWS::Region"
                  - AccountId
```

函数

您可以在配置文件中使用函数，通过来自其他资源或来自 Elastic Beanstalk 配置选项设置的信息填充资源属性的值。Elastic Beanstalk 支持 AWS CloudFormation 函数（`Ref`、`Fn::GetAtt`、`Fn::Join`）和一个 Elastic Beanstalk 特定函数 `Fn::GetOptionSetting`。

函数

- Ref
- Fn::GetAtt
- Fn::Join
- Fn::GetOptionSetting

Ref

使用 `Ref` 可检索 AWS 资源的默认字符串表示形式。`Ref` 返回的值取决于资源类型，有时也取决于其他因素。例如，安全组 (AWS::EC2::SecurityGroup) 将返回安全组的名称或 ID，这取决于安全组是位于默认 Amazon Virtual Private Cloud (Amazon VPC)、EC2 Classic 还是自定义 VPC 中。

```
{ "Ref" : "resource name" }
```

> **Note**
>
> 有关每个资源类型的详细信息（包括 `Ref` 的返回值），请参阅 AWS CloudFormation 用户指南中的 AWS资源类型参考。

在 Auto Scaling 生命周期挂钩示例中：

```
Resources:
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      AutoScalingGroupName: { "Ref" : "AWSEBAutoScalingGroup" }
```

您还可以使用 `Ref` 检索在同一文件的其他位置或在不同配置文件中定义的 AWS CloudFormation 参数的值。

Fn::GetAtt

使用 `Fn::GetAtt` 可检索 AWS 资源上的属性的值。

```
{ "Fn::GetAtt" : [ "resource name", "attribute name"] }
```

在 Auto Scaling 生命周期挂钩示例中：

```
Resources:
  lifecyclehook:
    Type: AWS::AutoScaling::LifecycleHook
    Properties:
      RoleARN: { "Fn::GetAtt" : [ "hookrole", "Arn"] }
```

有关更多信息，请参阅 Fn::GetAtt。

Fn::Join

使用 `Fn::Join` 可组合带有一个分隔符的字符串。这些字符串可以进行硬编码，也可以使用来自 `Fn::GetAtt` 或 `Ref` 的输出。

```
{ "Fn::Join" : [ "delimiter", [ "string1", "string2" ] ] }
```

有关更多信息，请参阅 Fn::Join。

Fn::GetOptionSetting

使用 Fn::GetOptionSetting 可检索应用于环境的配置选项设置的值。

```
"Fn::GetOptionSetting":
  Namespace: "namespace"
  OptionName: "option name"
  DefaultValue: "default value"
```

在存储私有密钥示例中：

```
Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-west-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"
```

自定义资源示例

以下是可以用于自定义 Elastic Beanstalk 环境的一系列示例配置文件：

- DynamoDB、CloudWatch 和 SNS
- Elastic Load Balancing 和 CloudWatch
- ElastiCache
- RDS 和 CloudWatch
- SQS、SNS 和 CloudWatch

本页的子主题提供了一些在 Elastic Beanstalk 环境中添加和配置自定义资源的扩展示例。

示例

- 示例：ElastiCache
- 示例：SQS、CloudWatch 和 SNS

- 示例：DynamoDB、CloudWatch 和 SNS

示例：ElastiCache

下列示例将一个 Amazon ElastiCache 集群分别添加到 EC2-Classic 和 EC2-VPC（默认和自定义 Amazon Virtual Private Cloud (Amazon VPC)）平台。有关这些平台以及如何确定对于您的区域和 AWS账户 EC2 支持哪些平台的更多信息，请参阅 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-supported-platforms.html。然后请参阅本主题中适用于您的平台的部分。

- EC2-classic 平台
- EC2-VPC (默认)
- EC2-VPC (自定义)

EC2-classic 平台

此示例将一个 Amazon ElastiCache 集群添加到在 EC2-Classic 平台中启动实例的环境。此示例中列出的所有属性是为每种资源类型必须设置的最低要求。您可以在 ElastiCache 示例中下载该示例。

> **Note**
>
> 本示例将创建AWS资源，您可能要为其付费。有关 AWS 定价的更多信息，请参阅 https://aws.amazon.com/pricing/。某些服务是AWS免费使用套餐的一部分。如果您是新客户，则可免费试用这些服务。参阅 https://aws.amazon.com/free/ 了解更多信息。

要使用此示例，请执行下列操作：

1. 在源包的顶级目录中创建 `.ebextensions` 目录。
2. 创建两个扩展名为 `.config` 的配置文件并将其放入您的 `.ebextensions` 目录。一个配置文件定义资源，另一个配置文件定义选项。
3. 将应用程序部署到 Elastic Beanstalk。

   YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

创建定义资源的配置文件 (例如，`elasticache.config`)。在此示例中，我们通过指定 ElastiCache 集群资源的名称 (`MyElastiCache`)，声明它的类型，然后配置集群的属性，从而创建 ElastiCache

集群。该示例引用在此配置文件中创建和定义的 ElastiCache 安全组资源的名称。接下来，我们会创建 ElastiCache 安全组。我们会定义这个资源的名称、声明它的类型和添加该安全组的描述。最后，我们设置 ElastiCache 安全组的进入规则，以便仅为 ElastiCache 安全组 (MyCacheSecurityGroup) 和 Elastic Beanstalk 安全组 (AWSEBSecurityGroup) 内的实例授予访问权限。参数名 AWSEBSecurityGroup 是 Elastic Beanstalk 提供的固定资源名称。您必须将 AWSEBSecurityGroup 添加到您的 ElastiCache 安全组传入规则中，以便 Elastic Beanstalk 应用程序可以连接到 ElastiCache 集群中的实例。

```
#This sample requires you to create a separate configuration file that defines the
 custom option settings for CacheCluster properties.

Resources:
  MyElastiCache:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      CacheNodeType:
         Fn::GetOptionSetting:
             OptionName : CacheNodeType
             DefaultValue: cache.m1.small
      NumCacheNodes:
           Fn::GetOptionSetting:
             OptionName : NumCacheNodes
             DefaultValue: 1
      Engine:
           Fn::GetOptionSetting:
             OptionName : Engine
             DefaultValue: memcached
      CacheSecurityGroupNames:
        - Ref: MyCacheSecurityGroup
  MyCacheSecurityGroup:
    Type: AWS::ElastiCache::SecurityGroup
    Properties:
      Description: "Lock cache down to webserver access only"
  MyCacheSecurityGroupIngress:
    Type: AWS::ElastiCache::SecurityGroupIngress
    Properties:
      CacheSecurityGroupName:
        Ref: MyCacheSecurityGroup
      EC2SecurityGroupName:
        Ref: AWSEBSecurityGroup
```

有关此示例配置文件中使用的资源的更多信息，请参阅以下参考：

- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::SecurityGroup
- AWS::ElastiCache:SecurityGroupIngress

创建名为 options.config 的单独配置文件，并定义自定义选项设置。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     CacheNodeType : cache.m1.small
     NumCacheNodes : 1
     Engine : memcached
```

这些行会指示 Elastic Beanstalk 从配置文件（本示例中为 options.config）的 CacheNodeType、NumCacheNodes 和 Engine 值中获取 CacheNodeType、NumCacheNodes 和 Engine 属性的值，该配置文件的 option_settings 部分带有 aws:elasticbeanstalk:customoption 部分，后者的名称-值对中包含了实际要使用的值。在以上示例中，这意味着 cache.m1.small、1 和 memcached 将用于这些值。有关 Fn::GetOptionSetting 的更多信息，请参阅 函数。

EC2-VPC (默认)

此示例将一个 Amazon ElastiCache 集群添加到在 EC2-VPC 平台中启动实例的环境。具体来说，此部分中的信息适用于 EC2 在默认 VPC 中启动实例的情景。此示例中的所有属性是为每种资源类型必须设置的最低要求。有关默认 VPC 的更多信息，请参阅默认 VPC 和子网。

> **Note**
>
> 本示例将创建AWS资源，您可能要为其付费。有关 AWS 定价的更多信息，请参阅 https://aws.amazon.com/pricing/。某些服务是AWS免费使用套餐的一部分。如果您是新客户，则可免费试用这些服务。参阅 https://aws.amazon.com/free/ 了解更多信息。

要使用此示例，请执行下列操作：

1. 在源包的顶级目录中创建 .ebextensions 目录。
2. 创建两个扩展名为 .config 的配置文件并将其放入您的 .ebextensions 目录。一个配置文件定义资源，另一个配置文件定义选项。
3. 将应用程序部署到 Elastic Beanstalk。

YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

现在将资源配置文件命名为 `elasticache.config`。为了创建 ElastiCache 集群，此示例指定 ElastiCache 集群资源的名称 (`MyElastiCache`)，声明它的类型，然后配置集群的属性。该示例引用在此配置文件中创建和定义的安全组资源的 ID。

接下来，我们会创建 EC2 安全组。我们定义此资源的名称，声明其类型，添加描述，并为安全组设置传入规则以便仅允许从 Elastic Beanstalk 安全组 (`AWSEBSecurityGroup`) 中的实例进行访问。（参数名称 `AWSEBSecurityGroup` 是 Elastic Beanstalk 提供的固定资源名称。您必须将 `AWSEBSecurityGroup` 添加到您的 ElastiCache 安全组传入规则中，以便 Elastic Beanstalk 应用程序可以连接到 ElastiCache 集群中的实例。）

EC2 安全组的传入规则还定义缓存节点可以用于接受连接的 IP 协议和端口号。对于 Redis，默认端口号是 6379。

```
#This sample requires you to create a separate configuration file that defines the
 custom option settings for CacheCluster properties.

Resources:
  MyCacheSecurityGroup:
    Type: "AWS::EC2::SecurityGroup"
    Properties:
      GroupDescription: "Lock cache down to webserver access only"
      SecurityGroupIngress :
        - IpProtocol : "tcp"
          FromPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          ToPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          SourceSecurityGroupName:
            Ref: "AWSEBSecurityGroup"
  MyElastiCache:
    Type: "AWS::ElastiCache::CacheCluster"
    Properties:
      CacheNodeType:
        Fn::GetOptionSetting:
```

```
          OptionName : "CacheNodeType"
          DefaultValue : "cache.t2.micro"
      NumCacheNodes:
        Fn::GetOptionSetting:
          OptionName : "NumCacheNodes"
          DefaultValue : "1"
      Engine:
        Fn::GetOptionSetting:
          OptionName : "Engine"
          DefaultValue : "redis"
      VpcSecurityGroupIds:
        -
          Fn::GetAtt:
            - MyCacheSecurityGroup
            - GroupId

Outputs:
  ElastiCache:
    Description : "ID of ElastiCache Cache Cluster with Redis Engine"
    Value :
      Ref : "MyElastiCache"
```

有关此示例配置文件中使用的资源的更多信息，请参阅以下参考：

- AWS::ElastiCache::CacheCluster
- AWS::EC2::SecurityGroup

接下来，将选项配置文件命名为 `options.config`，然后定义自定义选项设置。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
    CacheNodeType : cache.t2.micro
    NumCacheNodes : 1
    Engine : redis
    CachePort : 6379
```

这些行告知 Elastic Beanstalk 从配置文件（在我们的示例中为 `CacheNodeType`）中的 `NumCacheNodes`、`Engine`、`CachePort` 和 `CacheNodeType` 值获取 `NumCacheNodes`、`Engine`、`CachePort` 和 `options.config` 属性的值。该文件包含 `aws:elasticbeanstalk:customoption` 节 (在 `option_settings` 下)，其中含有要使用的实

际值的名称/值对。在前面的示例中，会对这些值使用 cache.t2.micro、1、redis 和 6379。有关 Fn::GetOptionSetting 的更多信息，请参阅 函数。

EC2-VPC (自定义)

如果您在 EC2-VPC 平台上创建自定义 VPC 并将它指定为 EC2 在其中启动实例的 VPC，则向环境添加 Amazon ElastiCache 集群的过程与默认 VPC 的过程不同。主要区别在于您必须为 ElastiCache 集群创建子网组。此示例中的所有属性是为每种资源类型必须设置的最低要求。

> **Note**
>
> 本示例将创建AWS资源，您可能要为其付费。有关 AWS 定价的更多信息，请参阅 https://aws.amazon.com/pricing/。某些服务是AWS免费使用套餐的一部分。如果您是新客户，则可免费试用这些服务。参阅 https://aws.amazon.com/free/ 了解更多信息。

要使用此示例，请执行下列操作：

1. 在源包的顶级目录中创建 .ebextensions 目录。
2. 创建两个扩展名为 .config 的配置文件并将其放入您的 .ebextensions 目录。一个配置文件定义资源，另一个配置文件定义选项。
3. 将应用程序部署到 Elastic Beanstalk。

   YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

现在将资源配置文件命名为 elasticache.config。为了创建 ElastiCache 集群，此示例指定 ElastiCache 集群资源的名称 (MyElastiCache)，声明它的类型，然后配置集群的属性。示例中的属性引用 ElastiCache 集群的子网组名称以及我们在此配置文件中创建并定义的安全组资源的 ID。

接下来，我们会创建 EC2 安全组。我们定义此资源的名称，声明其类型，添加描述和 VPC ID，并为安全组设置传入规则以便仅允许从 Elastic Beanstalk 安全组 (AWSEBSecurityGroup) 中的实例进行访问。（参数名称 AWSEBSecurityGroup 是 Elastic Beanstalk 提供的固定资源名称。您必须将 AWSEBSecurityGroup 添加到您的 ElastiCache 安全组传入规则中，以便 Elastic Beanstalk 应用程序可以连接到 ElastiCache 集群中的实例。）

EC2 安全组的传入规则还定义缓存节点可以用于接受连接的 IP 协议和端口号。对于 Redis，默认端口号是 6379。最后，此示例为 ElastiCache 集群创建子网组。我们定义此资源的名称，声明其类型，并在子网组中添加子网的描述和 ID。

> **Note**
>
> 我们建议您对 ElastiCache 集群使用私有子网。有关具有私有子网的 VPC 的更多信息，请参阅 https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html。

```
#This sample requires you to create a separate configuration file that defines the
 custom option settings for CacheCluster properties.

Resources:
  MyElastiCache:
    Type: "AWS::ElastiCache::CacheCluster"
    Properties:
      CacheNodeType:
        Fn::GetOptionSetting:
          OptionName : "CacheNodeType"
          DefaultValue : "cache.t2.micro"
      NumCacheNodes:
        Fn::GetOptionSetting:
          OptionName : "NumCacheNodes"
          DefaultValue : "1"
      Engine:
        Fn::GetOptionSetting:
          OptionName : "Engine"
          DefaultValue : "redis"
      CacheSubnetGroupName:
        Ref: "MyCacheSubnets"
      VpcSecurityGroupIds:
        - Ref: "MyCacheSecurityGroup"
  MyCacheSecurityGroup:
    Type: "AWS::EC2::SecurityGroup"
    Properties:
      GroupDescription: "Lock cache down to webserver access only"
      VpcId:
        Fn::GetOptionSetting:
          OptionName : "VpcId"
      SecurityGroupIngress :
        - IpProtocol : "tcp"
          FromPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
```

```
          ToPort :
            Fn::GetOptionSetting:
              OptionName : "CachePort"
              DefaultValue: "6379"
          SourceSecurityGroupId:
            Ref: "AWSEBSecurityGroup"
  MyCacheSubnets:
    Type: "AWS::ElastiCache::SubnetGroup"
    Properties:
      Description: "Subnets for ElastiCache"
      SubnetIds:
        Fn::GetOptionSetting:
          OptionName : "CacheSubnets"
Outputs:
  ElastiCache:
    Description : "ID of ElastiCache Cache Cluster with Redis Engine"
    Value :
      Ref : "MyElastiCache"
```

有关此示例配置文件中使用的资源的更多信息，请参阅以下参考：

- AWS::ElastiCache::CacheCluster
- AWS::EC2::SecurityGroup
- AWS::ElastiCache::SubnetGroup

接下来，将选项配置文件命名为 `options.config`，然后定义自定义选项设置。

> **Note**
>
> 在以下示例中，将示例 `CacheSubnets` 和 `VpcId` 值替换为自己的子网和 VPC。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
    CacheNodeType : cache.t2.micro
    NumCacheNodes : 1
    Engine : redis
    CachePort : 6379
    CacheSubnets:
      - subnet-1a1a1a1a
      - subnet-2b2b2b2b
```

```
      - subnet-3c3c3c3c
    VpcId: vpc-4d4d4d4d
```

这些行告知 Elastic Beanstalk 从配置文件（在我们的示例中为 CacheNodeType）中的 NumCacheNodes、Engine、CachePort、CacheSubnets、VpcId 和 CacheNodeType 值获取 NumCacheNodes、Engine、CachePort、CacheSubnets、VpcId 和 options.config 属性的值。该文件包含 aws:elasticbeanstalk:customoption 节 (在 option_settings 下)，其中含有具有示例值的名称/值对。在以上示例中，对这些值使用 cache.t2.micro、1、redis、6379、subnet-1a1a1a1a、subnet-2b2b2b2b、subnet-3c3c3c3c 和 vpc-4d4d4d4d。有关 Fn::GetOptionSetting 的更多信息，请参阅 函数。

示例：SQS、CloudWatch 和 SNS

此示例给环境添加 Amazon SQS 队列以及有关队列深度的警报。此示例中显示的属性是必须具备的最低程度的属性，必须为每个资源进行设置。您可以在 SQS、SNS 和 CloudWatch 下载该示例。

> **Note**
>
> 本示例将创建AWS资源，您可能要为其付费。有关 AWS 定价的更多信息，请参阅 https://aws.amazon.com/pricing/。某些服务是AWS免费使用套餐的一部分。如果您是新客户，则可免费试用这些服务。参阅 https://aws.amazon.com/free/ 了解更多信息。

要使用此示例，请执行下列操作：

1. 在源包的顶级目录中创建 .ebextensions 目录。
2. 创建两个扩展名为 .config 的配置文件并将其放入您的 .ebextensions 目录。一个配置文件定义资源，另一个配置文件定义选项。
3. 将应用程序部署到 Elastic Beanstalk。

   YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

创建定义资源的配置文件 (例如，sqs.config)。在此示例中，我们创建 SQS 队列并定义 VisbilityTimeout 资源中的 MySQSQueue 属性。下一步，我们创建 SNS Topic，并指定电子邮件在警报激发时发送到 someone@example.com。最后，我们创建 CloudWatch 警报，在队列消息超过 10 个时激发。在 Dimensions 属性中，我们指定维度的名称以及代表维度度量的值。我们使用 Fn::GetAtt 从 QueueName 中返回 MySQSQueue 的值。

```
#This sample requires you to create a separate configuration file to define the custom
 options for the SNS topic and SQS queue.
Resources:
  MySQSQueue:
    Type: AWS::SQS::Queue
    Properties:
      VisibilityTimeout:
        Fn::GetOptionSetting:
          OptionName: VisibilityTimeout
          DefaultValue: 30
  AlarmTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint:
            Fn::GetOptionSetting:
              OptionName: AlarmEmail
              DefaultValue: "nobody@amazon.com"
          Protocol: email
  QueueDepthAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: "Alarm if queue depth grows beyond 10 messages"
      Namespace: "AWS/SQS"
      MetricName: ApproximateNumberOfMessagesVisible
      Dimensions:
        - Name: QueueName
          Value : { "Fn::GetAtt" : [ "MySQSQueue", "QueueName"] }
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 1
      Threshold: 10
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: AlarmTopic
      InsufficientDataActions:
        - Ref: AlarmTopic

Outputs :
  QueueURL:
    Description : "URL of newly created SQS Queue"
    Value : { Ref : "MySQSQueue" }
  QueueARN :
```

```
    Description : "ARN of newly created SQS Queue"
    Value : { "Fn::GetAtt" : [ "MySQSQueue", "Arn"]}
  QueueName :
    Description : "Name newly created SQS Queue"
    Value : { "Fn::GetAtt" : [ "MySQSQueue", "QueueName"]}
```

有关此示例配置文件中使用的资源的更多信息，请参阅以下参考：

- Amazon::SQS::Queue
- AWS::SNS::Topic
- AWS::CloudWatch::Alarm

创建名为 options.config 的单独配置文件，并定义自定义选项设置。

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     VisibilityTimeout : 30
     AlarmEmail : "nobody@example.com"
```

这些行会指示 Elastic Beanstalk 从配置文件（本示例中为 options.config）的 VisibilityTimeout 和 Subscription Endpoint 值中获取 VisibilityTimeout 和 Subscription Endpoint 属性的值，该配置文件的 option_settings 部分带有 aws:elasticbeanstalk:customoption 部分，后者的名称-值对中包含了实际要使用的值。在以上示例中，这意味着 30 和“nobody@amazon.com”将用于这些值。有关 Fn::GetOptionSetting 的更多信息，请参阅 the section called “函数”。

示例：DynamoDB、CloudWatch 和 SNS

此配置文件会使用适用于 PHP 2 的 AWS 开发工具包将 DynamoDB 表设置为基于 PHP 的应用程序的会话处理程序。要使用此示例，您必须拥有 IAM 实例配置文件，该文件会添加到环境中的实例，并用来访问 DynamoDB 表。

您可以在 DynamoDB 会话支持示例中下载该示例，我们在本步骤会用到这个示例。此示例包含以下文件：

- 示例应用程序 index.php
- 配置文件 dynamodb.config，用于创建和配置 DynamoDB 表及其他 AWS 资源，以及在 Elastic Beanstalk 环境中托管该应用程序的 EC2 实例上安装软件
- 配置文件 options.config，该文件会使用此特定安装的特定设置覆盖 dynamodb.config 中的默认设置

index.php

```
<?php

// Include the SDK using the Composer autoloader
require '../vendor/autoload.php';

use Aws\DynamoDb\DynamoDbClient;

// Grab the session table name and region from the configuration file
list($tableName, $region) = file(__DIR__ . '/../sessiontable');
$tableName = rtrim($tableName);
$region = rtrim($region);

// Create a DynamoDB client and register the table as the session handler
$dynamodb = DynamoDbClient::factory(array('region' => $region));
$handler = $dynamodb->registerSessionHandler(array('table_name' => $tableName,
 'hash_key' => 'username'));

// Grab the instance ID so we can display the EC2 instance that services the request
$instanceId = file_get_contents("http://169.254.169.254/latest/meta-data/instance-id");
?>
<h1>Elastic Beanstalk PHP Sessions Sample</h1>
<p>This sample application shows the integration of the Elastic Beanstalk PHP
container and the session support for DynamoDB from the AWS SDK for PHP 2.
Using DynamoDB session support, the application can be scaled out across
multiple web servers. For more details, see the
<a href="https://aws.amazon.com/php/">PHP Developer Center</a>.</p>

<form id="SimpleForm" name="SimpleForm" method="post" action="index.php">
<?php
echo 'Request serviced from instance ' . $instanceId . '<br/>';
echo '<br/>';

if (isset($_POST['continue'])) {
  session_start();
  $_SESSION['visits'] = $_SESSION['visits'] + 1;
  echo 'Welcome back ' . $_SESSION['username'] . '<br/>';
  echo 'This is visit number ' . $_SESSION['visits'] . '<br/>';
  session_write_close();
  echo '<br/>';
  echo '<input type="Submit" value="Refresh" name="continue" id="continue"/>';
  echo '<input type="Submit" value="Delete Session" name="killsession"
 id="killsession"/>';
```

```
} elseif (isset($_POST['killsession'])) {
  session_start();
  echo 'Goodbye ' . $_SESSION['username'] . '<br/>';
  session_destroy();
  echo 'Username: <input type="text" name="username" id="username" size="30"/><br/>';
  echo '<br/>';
  echo '<input type="Submit" value="New Session" name="newsession" id="newsession"/>';
} elseif (isset($_POST['newsession'])) {
  session_start();
  $_SESSION['username'] = $_POST['username'];
  $_SESSION['visits'] = 1;
  echo 'Welcome to a new session ' . $_SESSION['username'] . '<br/>';
  session_write_close();
  echo '<br/>';
  echo '<input type="Submit" value="Refresh" name="continue" id="continue"/>';
  echo '<input type="Submit" value="Delete Session" name="killsession"
 id="killsession"/>';
} else {
  echo 'To get started, enter a username.<br/>';
  echo '<br/>';
  echo 'Username: <input type="text" name="username" id="username" size="30"/><br/>';
  echo '<input type="Submit" value="New Session" name="newsession" id="newsession"/>';
}
?>
</form>
```

.ebextensions/dynamodb.config

```
Resources:
  SessionTable:
    Type: AWS::DynamoDB::Table
    Properties:
      KeySchema:
        HashKeyElement:
          AttributeName:
            Fn::GetOptionSetting:
              OptionName : SessionHashKeyName
              DefaultValue: "username"
          AttributeType:
            Fn::GetOptionSetting:
              OptionName : SessionHashKeyType
              DefaultValue: "S"
      ProvisionedThroughput:
```

```
        ReadCapacityUnits:
          Fn::GetOptionSetting:
            OptionName : SessionReadCapacityUnits
            DefaultValue: 1
        WriteCapacityUnits:
          Fn::GetOptionSetting:
            OptionName : SessionWriteCapacityUnits
            DefaultValue: 1

  SessionWriteCapacityUnitsLimit:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: { "Fn::Join" : ["", [{ "Ref" : "AWSEBEnvironmentName" }, "
write capacity limit on the session table." ]]}
      Namespace: "AWS/DynamoDB"
      MetricName: ConsumedWriteCapacityUnits
      Dimensions:
        - Name: TableName
          Value: { "Ref" : "SessionTable" }
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 12
      Threshold:
          Fn::GetOptionSetting:
            OptionName : SessionWriteCapacityUnitsAlarmThreshold
            DefaultValue: 240
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: SessionAlarmTopic
      InsufficientDataActions:
        - Ref: SessionAlarmTopic

  SessionReadCapacityUnitsLimit:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: { "Fn::Join" : ["", [{ "Ref" : "AWSEBEnvironmentName" }, " read
capacity limit on the session table." ]]}
      Namespace: "AWS/DynamoDB"
      MetricName: ConsumedReadCapacityUnits
      Dimensions:
        - Name: TableName
          Value: { "Ref" : "SessionTable" }
      Statistic: Sum
      Period: 300
```

```
      EvaluationPeriods: 12
      Threshold:
          Fn::GetOptionSetting:
            OptionName : SessionReadCapacityUnitsAlarmThreshold
            DefaultValue: 240
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: SessionAlarmTopic
      InsufficientDataActions:
        - Ref: SessionAlarmTopic

  SessionThrottledRequestsAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: { "Fn::Join" : ["", [{ "Ref" : "AWSEBEnvironmentName" }, ":
requests are being throttled." ]]}
      Namespace: AWS/DynamoDB
      MetricName: ThrottledRequests
      Dimensions:
        - Name: TableName
          Value: { "Ref" : "SessionTable" }
      Statistic: Sum
      Period: 300
      EvaluationPeriods: 1
      Threshold:
        Fn::GetOptionSetting:
          OptionName: SessionThrottledRequestsThreshold
          DefaultValue: 1
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - Ref: SessionAlarmTopic
      InsufficientDataActions:
        - Ref: SessionAlarmTopic

  SessionAlarmTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint:
            Fn::GetOptionSetting:
              OptionName: SessionAlarmEmail
              DefaultValue: "nobody@amazon.com"
          Protocol: email
```

```
files:
  "/var/app/sessiontable":
    mode: "000444"
    content: |
      `{"Ref" : "SessionTable"}`
      `{"Ref" : "AWS::Region"}`

  "/var/app/composer.json":
    mode: "000744"
    content:
      {
        "require": {
           "aws/aws-sdk-php": "*"
        }
      }

container_commands:
 "1-install-composer":
   command: "cd /var/app; curl -s http://getcomposer.org/installer | php"
 "2-install-dependencies":
   command: "cd /var/app; php composer.phar install"
 "3-cleanup-composer":
   command: "rm -Rf /var/app/composer.*"
```

在示例配置文件中，我们首先创建 DynamoDB 表，并配置该表的主密钥结构和容量单位，以便分配足够的资源提供所要求的吞吐量。下一步，我们为 `WriteCapacity` 和 `ReadCapacity` 创建 CloudWatch 警报。我们会创建 SNS 主题，该主题会在警报阈值被突破时将电子邮件发送到“nobody@amazon.com”。

在为环境创建和配置 AWS 资源后，需要自定义 EC2 实例。我们使用 `files` 密钥将 DynamoDB 表的详细信息传递给环境中的 EC2 实例，并为适用于 PHP 2 的 AWS 开发工具包的 `composer.json` 文件添加“require”。最后，我们会运行容器命令安装编辑器和必需的依赖项，然后删除安装程序。

`.ebextensions/options.config`

```
option_settings:
  "aws:elasticbeanstalk:customoption":
     SessionHashKeyName                      : username
     SessionHashKeyType                      : S
     SessionReadCapacityUnits                : 1
     SessionReadCapacityUnitsAlarmThreshold  : 240
     SessionWriteCapacityUnits               : 1
     SessionWriteCapacityUnitsAlarmThreshold : 240
```

```
    SessionThrottledRequestsThreshold       : 1
    SessionAlarmEmail                       : me@example.com
```

使用您希望收到的警报通知的电子邮件地址取代 SessionAlarmEmail 值。options.config 文件包含 dynamodb.config 中用于定义的部分变量的值。例如，dynamodb.config 包含以下行：

```
Subscription:
  - Endpoint:
      Fn::GetOptionSetting:
        OptionName: SessionAlarmEmail
        DefaultValue: "nobody@amazon.com"
```

这些行会指示 Elastic Beanstalk 从配置文件（示例应用程序中为 options.config）的 SessionAlarmEmail 值中获取 Endpoint 属性的值，该配置文件的 option_settings 部分带有 aws:elasticbeanstalk:customoption 部分，后者的名称-值对中包含了实际要使用的值。在以上示例中，这意味着 SessionAlarmEmail 将被分配 nobody@amazon.com 值。

有关此示例中使用的 CloudFormation 资源的详细信息，请参阅以下参考：

- AWS::DynamoDB::Table
- AWS::CloudWatch::Alarm
- AWS::SNS::Topic

使用 Elastic Beanstalk 保存的配置

您可将环境的配置作为对象保存在 Amazon Simple Storage Service (Amazon S3) 中，以便在创建环境时应用于其他环境，或应用于正在运行的环境。保存的配置 是 YAML 格式的模板，用于定义环境的平台版本、层、配置选项设置和标记。

您可以在创建保存的配置和编辑现有保存的配置的标签时向其应用标签。应用到保存的配置的标签与使用 Tags: 键在保存的配置中指定的标签无关。当您将保存的配置应用到环境时，后者将应用到环境。有关详细信息，请参阅 标记保存的配置。

> **Note**
>
> 您可以使用多种方法创建保存的配置并将其应用于您的 Elastic Beanstalk 环境。其中包括 Elastic Beanstalk 控制台、EB CLI 和 AWS CLI。
> 有关创建和应用保存的配置的替代方法示例，请参阅下列主题：

- 在创建环境之前设置配置选项
- 在环境创建期间设置配置选项
- 在环境创建后设置配置选项

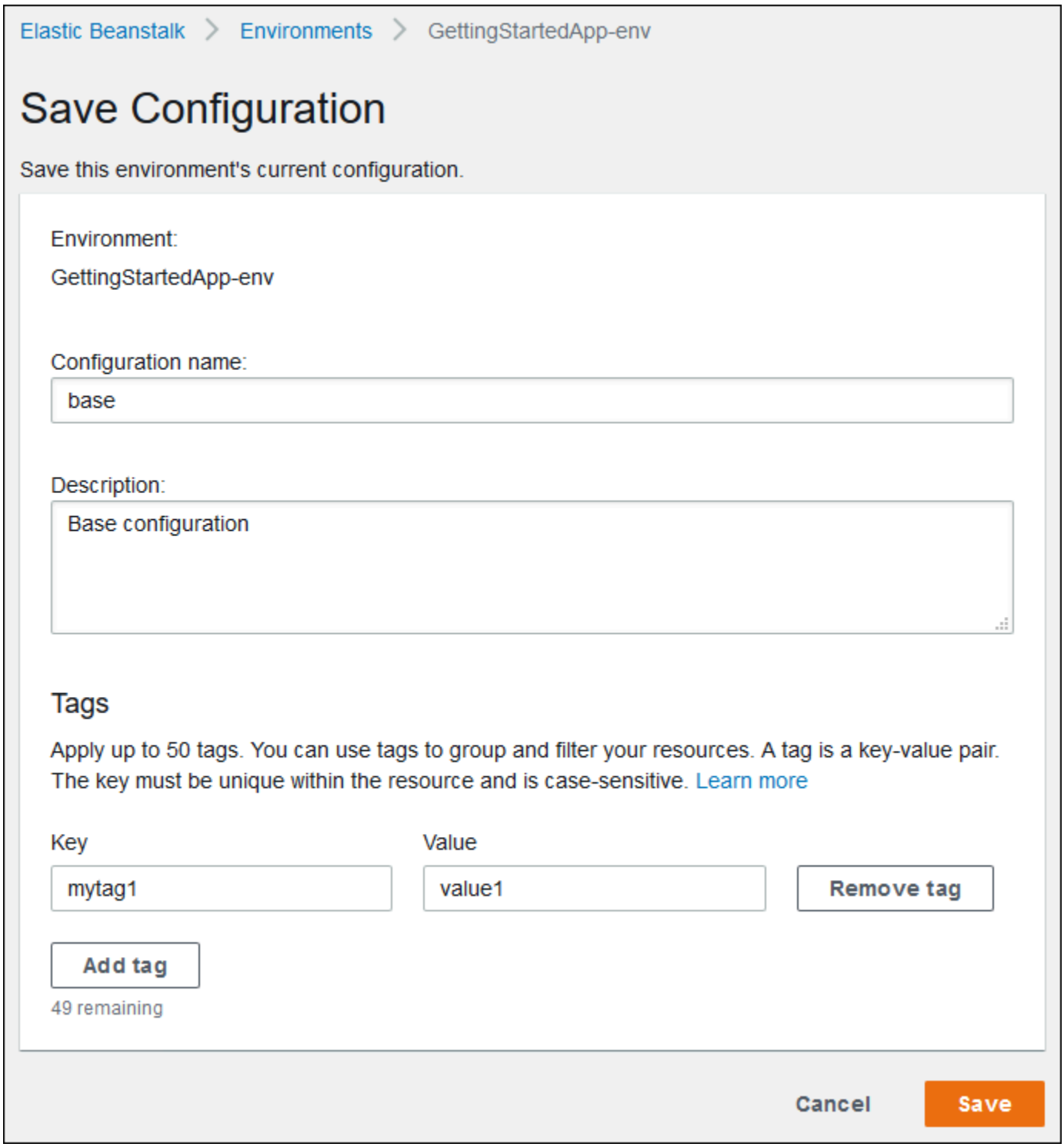
在 Elastic Beanstalk 管理控制台中，从环境的当前状态创建保存的配置。

保存环境的配置

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Save configuration（保存配置）。
4. 使用屏幕上的表单命名已保存的配置。（可选）提供简要说明，并添加标签键和值。
5. 选择 Save（保存）。



保存的配置包括已应用于具有控制台的环境或任何其他使用 Elastic Beanstalk API 的客户端的任何设置。您可以稍后将保存的配置应用于您的环境以将它还原为上一个状态，也可以在创建环境期间将保存的配置应用于新环境。

您可以使用 EB CLI the section called “eb config” 命令下载配置，如以下示例所示。*NAME* 是保存的配置的名称。

```
eb config get NAME
```

在创建环境期间应用保存的配置（Elastic Beanstalk 控制台）

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Saved configurations（已保存的配置）。
4. 选择要应用的已保存的配置，然后选择 Launch environment（启动环境）。
5. 继续执行向导以创建环境。

保存的配置不包括已与应用程序源代码中的配置文件一起应用的设置。如果在配置文件和保存的配置中应用相同的设置，则保存的配置中的设置优先。同样，Elastic Beanstalk 控制台中指定的选项将覆盖保存的配置中的选项。有关更多信息，请参阅优先顺序。

保存的配置存储在 Elastic Beanstalk S3 存储桶中按照应用程序命名的文件夹中。例如，对于账号 123456789012，us-west-2 区域中名为 `my-app` 的应用程序的配置位于 `s3://elasticbeanstalk-us-west-2-123456789012/resources/templates/my-app/`。

通过在文本编辑器中打开保存的配置，可以查看其内容。以下示例配置显示使用 Elastic Beanstalk 管理控制台启动的 Web 服务器环境的配置。

```
EnvironmentConfigurationMetadata:
  Description: Saved configuration from a multicontainer Docker environment created
 with the Elastic Beanstalk Management Console
  DateCreated: '1520633151000'
  DateModified: '1520633151000'
Platform:
  PlatformArn: arn:aws:elasticbeanstalk:us-east-2::platform/Java 8 running on 64bit
 Amazon Linux/2.5.0
OptionSettings:
  aws:elasticbeanstalk:command:
```

```
    BatchSize: '30'
    BatchSizeType: Percentage
  aws:elasticbeanstalk:sns:topics:
    Notification Endpoint: me@example.com
  aws:elb:policies:
    ConnectionDrainingEnabled: true
    ConnectionDrainingTimeout: '20'
  aws:elb:loadbalancer:
    CrossZone: true
  aws:elasticbeanstalk:environment:
    ServiceRole: aws-elasticbeanstalk-service-role
  aws:elasticbeanstalk:application:
    Application Healthcheck URL: /
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
  aws:autoscaling:launchconfiguration:
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    InstanceType: t2.micro
    EC2KeyName: workstation-uswest2
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Health
    RollingUpdateEnabled: true
EnvironmentTier:
  Type: Standard
  Name: WebServer
AWSConfigurationTemplateVersion: 1.1.0.0
Tags:
  Cost Center: WebApp Dev
```

您可修改保存的配置的内容并将其保存在 Amazon S3 中的同一位置。存储在正确位置的任何格式正确的保存的配置均可使用 Elastic Beanstalk 管理控制台应用于环境。

支持以下键。

- AWSConfigurationTemplateVersion（必需）– 配置模板版本 (1.1.0.0)。

```
AWSConfigurationTemplateVersion: 1.1.0.0
```

- Platform – 环境的平台版本的 Amazon 资源名称 (ARN)。您可以按 ARN 或解决方案堆栈名称指定平台。

```
Platform:
```

```
  PlatformArn: arn:aws:elasticbeanstalk:us-east-2::platform/Java 8 running on 64bit
Amazon Linux/2.5.0
```

- SolutionStack – 用于创建环境的解决方案堆栈的全名。

```
SolutionStack: 64bit Amazon Linux 2017.03 v2.5.0 running Java 8
```

- OptionSettings - 要应用到环境的配置选项设置。例如，下面的条目将实例类型设为 t2.micro。

```
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: t2.micro
```

- Tags - 多达 47 个标签，应用于在环境中创建的资源。

```
Tags:
  Cost Center: WebApp Dev
```

- EnvironmentTier - 要创建的环境的类型。对于 Web 服务器环境，您可不包含此部分 (Web 服务器为默认值)。对于工作线程环境，请使用以下设置。

```
EnvironmentTier:
  Name: Worker
  Type: SQS/HTTP
```

> **Note**
>
> 您可以使用多种方法创建保存的配置并将其应用于您的 Elastic Beanstalk 环境。其中包括 Elastic Beanstalk 控制台、EB CLI 和 AWS CLI。
> 有关创建和应用保存的配置的替代方法示例，请参阅下列主题：
>
> - 在创建环境之前设置配置选项
> - 在环境创建期间设置配置选项
> - 在环境创建后设置配置选项

标记保存的配置

您可以将标签应用到 AWS Elastic Beanstalk 保存的配置。标签是与AWS资源关联的键/值对。有关 Elastic Beanstalk 资源标记、使用案例、标签键和值约束以及支持的资源类型的信息，请参阅标记 Elastic Beanstalk 应用程序资源。

您可以在创建保存的配置时指定标签。在现有保存的配置中，您可以添加或删除标签，以及更新现有标签的值。您最多可以为每个保存的配置添加 50 个标签。

在创建保存的配置期间添加标签

在使用 Elastic Beanstalk 控制台保存配置时，您可以在 Save Configuration (保存配置) 页面上指定标签键和值。

如果使用 EB CLI 保存配置，则可以使用 eb config 的 `--tags` 选项添加标签。

```
~/workspace/my-app$ eb config --tags mytag1=value1,mytag2=value2
```

对于 AWS CLI 或其他基于 API 的客户端，可以使用 create-configuration-template 命令的 `--tags` 参数添加标签。

```
$ aws elasticbeanstalk create-configuration-template \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --application-name my-app --template-name my-template --solution-stack-
name solution-stack
```

管理现有保存的配置的标签

可以在现有的 Elastic Beanstalk 保存的配置中添加、更新和删除标签。

使用 Elastic Beanstalk 控制台管理保存的配置的标签

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Applications（应用程序），然后从列表中选择应用程序的名称。

> **Note**
>
> 如果您有多个应用程序，请使用搜索栏筛选应用程序列表。

3. 在导航窗格中，找到应用程序的名称，然后选择 Saved configurations（已保存的配置）。
4. 选择要管理的已保存配置。
5. 选择 Actions (操作)，然后选择 Manage tags (管理标签)。
6. 使用屏幕上的表单添加、更新或删除标签。
7. 要保存更改，请选择页面底部的 Apply（应用）。

如果使用 EB CLI 更新保存的配置，则可使用 eb tags 来添加、更新、删除或列出标签。

例如，以下命令会列出保存的配置中的标签。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

以下命令会更新标签 `mytag1` 并删除标签 `mytag2`。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

有关选项和更多示例的完整列表，请参阅 `eb tags`。

对于 AWS CLI 或其他基于 API 的客户端，可使用 list-tags-for-resource 命令列出保存的配置的标签。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

使用 update-tags-for-resource 命令可在保存的配置中添加、更新或删除标签。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:configurationtemplate/my-app/my-template"
```

在 `--tags-to-add` 的 update-tags-for-resource 参数中指定要添加的标签和要更新的标签。添加了一个不存在的标签，更新了现有标签的值。

> **Note**
>
> 要将某些 EB CLI 和 AWS CLI 命令与 Elastic Beanstalk 保存的配置结合使用，您需要保存的配置的 ARN。要构造 ARN，首先请使用以下命令检索保存的配置的名称。
>
> ```
> $ aws elasticbeanstalk describe-applications --application-names my-app
> ```
>
> 在命令输出中查找 `ConfigurationTemplates` 键。此元素显示保存的配置的名称。在本页面上提到的命令中指定 *my-template* 的地方使用此名称。

环境清单 (`env.yaml`)

您可以在应用程序源包的根目录中包含一个 YAML 格式的环境清单，以配置在创建环境时使用的环境名称、解决方案堆栈和环境链接。

此文件格式包含对环境组的支持。要使用组，请在清单中指定环境名称并在末尾添加一个 + 号。创建或更新环境时，请使用 `--group-name` (AWS CLI) 或 `--env-group-suffix` (EB CLI) 指定组名称。有关组的更多信息，请参阅创建和更新 Elastic Beanstalk 环境组。

以下示例清单定义一个 Web 服务器环境，并包含一个指向它依赖的工作线程环境组件的链接。此清单使用组以允许使用相同的源包创建多个环境：

`~/myapp/frontend/env.yaml`

```
AWSConfigurationTemplateVersion: 1.1.0.0
SolutionStack: 64bit Amazon Linux 2015.09 v2.0.6 running Multi-container Docker 1.7.1
 (Generic)
OptionSettings:
  aws:elasticbeanstalk:command:
    BatchSize: '30'
    BatchSizeType: Percentage
  aws:elasticbeanstalk:sns:topics:
    Notification Endpoint: me@example.com
  aws:elb:policies:
    ConnectionDrainingEnabled: true
    ConnectionDrainingTimeout: '20'
  aws:elb:loadbalancer:
    CrossZone: true
  aws:elasticbeanstalk:environment:
    ServiceRole: aws-elasticbeanstalk-service-role
```

```
  aws:elasticbeanstalk:application:
    Application Healthcheck URL: /
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
  aws:autoscaling:launchconfiguration:
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
    InstanceType: t2.micro
    EC2KeyName: workstation-uswest2
  aws:autoscaling:updatepolicy:rollingupdate:
    RollingUpdateType: Health
    RollingUpdateEnabled: true
Tags:
  Cost Center: WebApp Dev
CName: front-A08G28LG+
EnvironmentName: front+
EnvironmentLinks:
  "WORKERQUEUE" : "worker+"
```

支持以下键。

- AWSConfigurationTemplateVersion（必需）– 配置模板版本 (1.1.0.0)。

```
AWSConfigurationTemplateVersion: 1.1.0.0
```

- Platform – 环境的平台版本的 Amazon 资源名称 (ARN)。您可以按 ARN 或解决方案堆栈名称指定平台。

```
Platform:
  PlatformArn: arn:aws:elasticbeanstalk:us-east-2::platform/Java 8 running on 64bit
 Amazon Linux/2.5.0
```

- SolutionStack – 用于创建环境的解决方案堆栈的全名。

```
SolutionStack: 64bit Amazon Linux 2017.03 v2.5.0 running Java 8
```

- OptionSettings - 要应用到环境的配置选项设置。例如，下面的条目将实例类型设为 t2.micro。

```
OptionSettings:
  aws:autoscaling:launchconfiguration:
    InstanceType: t2.micro
```

- Tags - 多达 47 个标签，应用于在环境中创建的资源。

```
Tags:
  Cost Center: WebApp Dev
```

- EnvironmentTier - 要创建的环境的类型。对于 Web 服务器环境，您可不包含此部分 (Web 服务器为默认值)。对于工作线程环境，请使用以下设置。

```
EnvironmentTier:
  Name: Worker
  Type: SQS/HTTP
```

- CName – 环境的别名记录。在名称末尾包含一个 + 号以启用组。

```
CName: front-A08G28LG+
```

- EnvironmentName – 要创建的环境的名称。在名称末尾包含一个 + 号以启用组。

```
EnvironmentName: front+
```

  如果启用组，则在创建环境时必须指定组名称。Elastic Beanstalk 使用连字符将组名称追加到环境名称。例如，对于环境名称 `front+` 和组名称 `dev`，Elastic Beanstalk 会创建名为 `front-dev` 的环境。

- EnvironmentLinks – 变量名称与依赖项环境名称的映射。下面的示例使 `worker+` 环境成为依赖项并告知 Elastic Beanstalk 将链接信息保存到名为 `WORKERQUEUE` 的变量。

```
EnvironmentLinks:
  "WORKERQUEUE" : "worker+"
```

  链接变量的值因被链接环境的类型而异。对于 Web 服务器环境，链接为环境的别名记录。对于工作线程环境，链接为环境的 Amazon Simple Queue Service (Amazon SQS) 队列的名称。

CName、EnvironmentName 和 EnvironmentLinks 键可用于创建环境组和指向其他环境的链接。在使用 EB CLI、AWS CLI 或软件开发工具包时，这些功能当前受支持。

使用自定义 Amazon 系统映像 (AMI)

创建 AWS Elastic Beanstalk 环境时，您可以指定要使用的亚马逊系统映像 (AMI)，而不是平台版本中包含的标准 Elastic Beanstalk AMI。在您的环境中启动实例时，如果您需要安装标准 AMI 未包含的大量软件，则使用自定义 AMI 可以缩短配置时间。

使用配置文件能够快速、一致地配置和自定义您的环境。但在环境创建和更新过程中，应用配置可能需要花费很长的时间。如果您需要在配置文件中进行大量的服务器配置，可以制作一份包含所需软件和配置的自定义 AMI，以缩短配置耗时。

此外，您还可以借助自定义 AMI 对底层组件 (如 Linux 内核) 进行更改，这在配置文件中很难实现或需要很长时间才能完成。要创建自定义 AMI，请在 Amazon EC2 中启动 Elastic Beanstalk 平台 AMI，根据需要自定义软件和配置，然后停止该实例并据之保存一个 AMI。

创建自定义 AMI

识别基本 Elastic Beanstalk AMI

1. 在命令窗口中，运行以下命令。有关更多信息，请参阅《AWS CLI 命令参考》describe-platform-version中的。

   指定您要使用自定义 AMI 的 AWS 区域，并将平台 ARN 和版本号替换为您的应用程序所基于的 Elastic Beanstalk 平台。

   Example - Mac OS / Linux OS

```
$ aws elasticbeanstalk describe-platform-version --region us-east-2 \
      --platform-arn "arn:aws:elasticbeanstalk:us-east-2::platform/Tomcat 8.5 with
 Java 8 running on 64bit Amazon Linux/3.1.6" \
      --query PlatformDescription.CustomAmiList
[
    {
        "VirtualizationType": "pv",
        "ImageId": ""
    },
    {
        "VirtualizationType": "hvm",
        "ImageId": "ami-020ae06fdda6a0f66"
    }
]
```

Example - Windows OS

```
C:\> aws elasticbeanstalk describe-platform-version --region us-east-2 --platform-
arn"arn:aws:elasticbeanstalk:us-east-2::platform/IIS 10.0 running on 64bit Windows
 Server 2019/2.6.4" --query PlatformDescription.CustomAmiList
[
    {
        "VirtualizationType": "pv",
        "ImageId": ""
    },
    {
        "VirtualizationType": "hvm",
        "ImageId": "ami-020ae06fdda6a0f66"
    }
]
```

2. 记下结果中类似于 ami-020ae06fdda6a0f66 的 ImageId 值。

该值是与您的应用程序相关的平台版本、EC2 实例架构 AWS 和区域的现货 Elastic Beanstalk AMI。如果您需要为多个平台、架构或 AWS 区域创建 AMI，请重复此过程，为每种组合确定正确的基础 AMI。

> **注意**
>
> - 不要从已在 Elastic Beanstalk 环境中启动的实例创建 AMI。Elastic Beanstalk 会在配置期间修改实例，这可能导致所保存的 AMI 出现问题。从 Elastic Beanstalk 环境中的实例保存映像还会使此实例上部署的应用程序版本成为映像的固定部分。
> - 我们建议您始终使用最新的平台版本。当您更新到新平台版本时，我们还建议您将自定义 AMI 变基为新平台版本的 AMI。这样可以最大限度地减少因软件包或库版本不兼容而导致的部署失败。

对于 Linux，也可以从并非由 Elastic Beanstalk 发布的社区 AMI 创建自定义 AMI。您可以使用最新的 Amazon Linux AMI 作为起点。使用不由 Elastic Beanstalk 管理的 Linux AMI 启动环境时，Elastic Beanstalk 会尝试安装平台软件（语言、框架、代理服务器等）及其他组件，以支持增强型运行状况报告等功能。

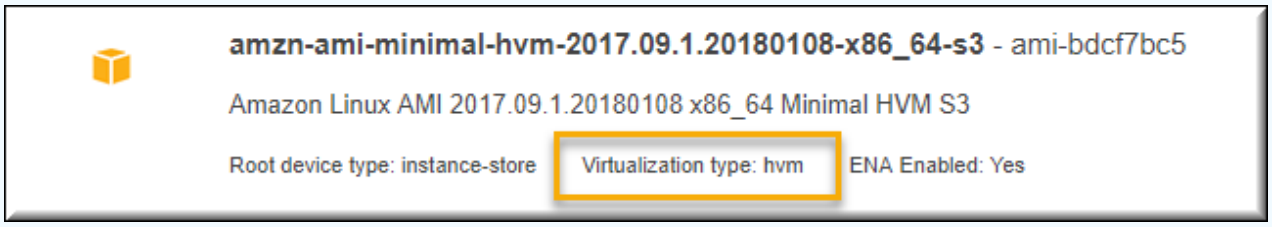
> **Note**
>
> 基于 Windows Server 的自定义 AMI 需要从 `describe-platform-version` 中返回库存 Elastic Beanstalk AMI，如前面的步骤 1 所示。

虽然 Elastic Beanstalk 可以使用不由 Elastic Beanstalk 管理的 AMI，但 Elastic Beanstalk 安装缺失的组件会导致配置时间增加，因而减少或抵消当初创建自定义 AMI 的优势。其他 Linux 发行版也许能够进行某些故障诊断操作，但不受官方支持。如果您的应用程序需要特定的 Linux 发行版，一个替代方案是创建 Docker 映像并在 Docker 平台或多容器 Docker 平台上运行该映像。

创建自定义 AMI

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择 Launch Instance（启动实例）。
3. 选择社区 AMI。
4. 如果您确定了基础 Elastic Beanstalk AMI（使用 `describe-platform-version`）或 Amazon Linux AMI，请在搜索框中输入其 AMI ID。然后按 Enter。

   您也可以搜索满足需要的其他社区 AMI 的列表。

   > **Note**
   >
   > 我们建议您选择使用 HVM 虚拟化的 AMI。这些 AMI 的描述包含 Virtualization type: hvm (虚拟化类型: hvm)。
   >
   > 
   >
   >
   > 有关实例虚拟化类型的详细信息，请参阅亚马逊 EC2 用户指南中的 Linux AM I 虚拟化类型或亚马逊 EC 2 用户指南中的 Windows AMI 虚拟化类型。

5. 选择 Select (选择) 以选择此 AMI。
6. 选择实例类型，然后选择下一步：配置实例详细信息。
7. （Linux 平台）展开高级详细信息部分，然后在用户数据字段中粘贴以下文本：

```
#cloud-config
  repo_releasever: repository version number
  repo_upgrade: none
```

存储库版本号 是指 AMI 名称中的年份和月份版本。例如，基于 Amazon Linux 2015 年 3 月版的 AMI 的存储库版本号为 2015.03。对于 Elastic Beanstalk 映像，该版本号是基于 Amazon Linux AMI（在 Amazon Linux 2 之前）的平台版本的解决方案堆栈名称中显示的日期。

> **Note**
>
> 该repo_releasever设置为亚马逊 Linux AMI 配置该 lock-on-launch 功能。这会导致 AMI 在启动时固定使用特定的存储库版本。Amazon Linux 2 不支持此功能 - 如果您的环境使用最新的 Amazon Linux 2 平台分支，请不要指定此功能。如果您仅在 Amazon Linux AMI 平台分支（在 Amazon Linux 2 之前）上将自定义 AMI 与 Elastic Beanstalk 结合使用，则需要此设置。
> repo_upgrade 设置会禁止自动安装安全更新。它需要将自定义 AMI 与 Elastic Beanstalk 结合使用。

8. 根据向导指示进行操作，以启动此 EC2 实例。当系统提示时，选择您能够访问的密钥对，以便能够连接此实例来执行后续步骤。
9. 使用 SSH 或 RDP 连接到此实例。
10. 执行任何所需的自定义操作。
11. （Windows 平台）运行 EC2Config 服务 Sysprep。有关 EC2Config 的信息，请参阅使用 EC2Config 服务配置 Windows 实例。确保 Sysprep 配置为生成可从 AWS Management Console 检索的随机密码。
12. 在 Amazon EC2 控制台中，停止 EC2 实例。然后，在 Instance Actions (实例操作) 菜单上，选择 Create Image (EBS AMI) (创建映像(EBS AMI))。
13. 为避免产生额外 AWS 费用，请终止 EC2 实例。

在 Elastic Beanstalk 环境中使用您的自定义 AMI

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration (配置)。
4. 在 Capacity (容量) 配置类别中，选择 Edit (编辑)。
5. 对于 AMI ID，请输入您的自定义 AMI ID。
6. 要保存更改，请选择页面底部的 Apply（应用）。

当您使用自定义 AMI 创建新环境时，应该使用您用作创建 AMI 的基础的相同平台版本。如果以后您使用自定义 AMI 将平台更新应用于环境，Elastic Beanstalk 会尝试在引导过程中应用库和配置更新。

清除自定义 AMI

在您使用完自定义 AMI 并且不再需要它来启动 Elastic Beanstalk 环境时，请考虑将其清除以最大程度地减少存储成本。清除自定义 AMI 涉及到从 Amazon EC2 取消注册它并删除其他关联的资源。有关详细信息，请参阅取消注册您的 Linux AMI 或取消注册您的 Windows AMI。

保持能够访问适用于已停用平台的亚马逊机器映像（AMI）的方法

当分支使用的操作系统或主要组件达到生命周期终点时，Elastic Beanstalk 将平台分支状态设置为停用。平台分支的基础 Elastic Beanstalk AMI 也可以私有化，以防止使用过时的 AMI。使用已私有化的 AMI 的环境将无法再启动实例。

如果无法在应用程序停用之前将其迁移到受支持的环境，则您的环境可能就会出现这种情况。可能需要更新 Beanstalk 平台分支的环境，在这个环境中，基础 Elastic Beanstalk AMI 已私有化。可以采用另一种方法。您可以根据环境使用的基础 Elastic Beanstalk AMI 副本更新现有环境。

本主题提供了一些步骤和一个独立脚本，用于根据环境使用的基础 Elastic Beanstalk AMI 副本更新现有环境。一旦将应用程序迁移到受支持的平台，您就可以继续使用标准过程来维护您的应用程序和受支持的环境。

手动步骤

要基于基础 Elastic Beanstalk AMI 的 AMI 副本更新环境

1. 确定环境使用的 AMI。此命令返回您在参数中提供的 Elastic Beanstalk 环境使用的 AMI。返回的值将在下一步中用作 source-ami-id。

在命令窗口中，运行以下命令。有关更多信息，请参阅《AWS CLI 命令参考》中的 describe-configuration-settings。

指定存储您要复制的源 AMI 的 AWS 区域。将应用程序名称和环境名称替换为基于源 AMI 的名称。输入查询参数的文本，如下所示。

Example

```
>aws elasticbeanstalk describe-configuration-settings \
  --application-name my-application \
  --environment-name my-environment \
  --region us-east-2 \
  --query "ConfigurationSettings[0].OptionSettings[?OptionName=='ImageId'] |
 [0].Value"
```

2. 将 AMI 复制到您的账户。此命令将返回新的 AMI，该 AMI 是通过复制上一步中返回的 source-ami-id 得到的。

> **Note**
>
> 请务必记下此命令输出的新 AMI id。您需要在下一步中输入此 id，替换示例命令中的 copied-ami-id。

在命令窗口中，运行以下命令。有关更多信息，请参见《AWS CLI 命令参考》中的 copy-image。

指定要复制的源 AMI 的 AWS 区域（--source-region）和要使用新的自定义 AMI 的区域（--region）。将 source-ami-id 替换为您要复制的映像的 AMI。source-ami-id 是由上一步中的命令返回的。将 new-ami-name 替换为描述目标区域中的新 AMI 的名称。遵循此过程的脚本通过将字符串 "Copy of" 附加到 source-ami-id 名称的开头来生成新的 AMI 名称。

```
>aws ec2 copy-image \
    --region us-east-2 \
    --source-image-id source-ami-id \
    --source-region us-east-2 \
    --name new-ami-name
```

3. 更新环境以使用复制的 AMI。该命令运行后，将返回环境的状态。

在命令窗口中，运行以下命令。有关更多信息，请参阅《AWS CLI 命令参考》中的 update-environment。

指定需要更新的环境和应用程序的 AWS 区域。将应用程序名称和环境名称替换为需要与上一步中的 copied-ami-id 关联的名称。对于 --option-setttings 参数，请将 *copied-ami-id* 替换为在上一条命令输出中记录的 AMI id。

```
>aws elasticbeanstalk update-environment \
  --application-name my-application \
  --environment-name my-environment \
  --region us-east-2 \
  --option-settings
 "Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId,Value=copied-ami-id"
```

> **Note**
>
> 为了最大限度地降低存储成本，当您不再需要它来启动 Elastic Beanstalk 环境时，可以考虑清理您的自定义 AMI。有关更多信息，请参阅清除自定义 AMI。

独立脚本

以下脚本提供了与前面的手动步骤相同的结果。选择以下链接下载脚本：copy_ami_and_update_env.zip。

脚本源：copy_ami_and_update_env.sh

```
#!/bin/bash

set -ue

USAGE="This script is used to copy an AMI used by your Elastic Beanstalk environment
 into your account to use in your environment.\n\n"
USAGE+="Usage:\n\n"
USAGE+="./$(basename $0) [OPTIONS]\n"
USAGE+="OPTIONS:\n"
USAGE+="\t--application-name <application-name>\tThe name of your Elastic Beanstalk
 application.\n"
```

```
USAGE+="\t--environment-name <environment-name>\tThe name of your Elastic Beanstalk
 environment.\n"
USAGE+="\t--region <region> \t\t\tThe AWS region your Elastic Beanstalk environment is
 deployed to.\n"
USAGE+="\n\n"
USAGE+="Script Usage Example(s):\n"
USAGE+="./$(basename $0) --application-name my-application --environment-name my-
environment --region us-east-1\n"

if [ $# -eq 0 ]; then
  echo -e $USAGE
  exit
fi

while [[ $# -gt 0 ]]; do
  case $1 in
    --application-name)       APPLICATION_NAME="$2"; shift ;;
    --environment-name)       ENVIRONMENT_NAME="$2"; shift ;;
    --region)                 REGION="$2"; shift ;;
    *)                        echo "Unknown option $1" ; echo -e $USAGE ; exit ;;
  esac
  shift
done

aws_cli_version="$(aws --version)"
if [ $? -ne 0 ]; then
  echo "aws CLI not found. Please install it: https://docs.aws.amazon.com/cli/latest/
userguide/getting-started-install.html. Exiting."
  exit 1
fi
echo "Using aws CLI version: ${aws_cli_version}"

account=$(aws sts get-caller-identity --query "Account" --output text)
echo "Using account ${account}"

environment_ami_id=$(aws elasticbeanstalk describe-configuration-settings \
  --application-name "$APPLICATION_NAME" \
  --environment-name "$ENVIRONMENT_NAME" \
  --region "$REGION" \
  --query "ConfigurationSettings[0].OptionSettings[?OptionName=='ImageId'] | [0].Value"
 \
  --output text)
echo "Image associated with environment ${ENVIRONMENT_NAME} is ${environment_ami_id}"
```

```
owned_image=$(aws ec2 describe-images \
  --owners self \
  --image-ids "$environment_ami_id" \
  --region "$REGION" \
  --query "Images[0]" \
  --output text)
if [ "$owned_image" != "None" ]; then
  echo "${environment_ami_id} is already owned by account ${account}. Exiting."
  exit
fi

source_image_name=$(aws ec2 describe-images \
  --image-ids "$environment_ami_id" \
  --region "$REGION" \
  --query "Images[0].Name" \
  --output text)
if [ "$source_image_name" = "None" ]; then
  echo "Cannot find ${environment_ami_id}. Please contact AWS support if you need
 additional help: https://aws.amazon.com/support."
  exit 1
fi

copied_image_name="Copy of ${source_image_name}"
copied_ami_id=$(aws ec2 describe-images \
  --owners self \
  --filters Name=name,Values="${copied_image_name}" \
  --region "$REGION" \
  --query "Images[0].ImageId" \
  --output text)
if [ "$copied_ami_id" != "None" ]; then
  echo "Detected that ${environment_ami_id} has already been copied by account
 ${account}. Skipping image copy."
else
  echo "Copying ${environment_ami_id} to account ${account} with name
 ${copied_image_name}"
  copied_ami_id=$(aws ec2 copy-image \
    --source-image-id "$environment_ami_id" \
    --source-region "$REGION" \
    --name "$copied_image_name" \
    --region "$REGION" \
    --query "ImageId" \
    --output text)
  echo "New AMI ID is ${copied_ami_id}"
```

```
  echo "Waiting for ${copied_ami_id} to become available"
  aws ec2 wait image-available \
    --image-ids "$copied_ami_id" \
    --region "$REGION"
  echo "${copied_ami_id} is now available"
fi

echo "Updating environment ${ENVIRONMENT_NAME} to use ${copied_ami_id}"
environment_status=$(aws elasticbeanstalk update-environment \
  --application-name "$APPLICATION_NAME" \
  --environment-name "$ENVIRONMENT_NAME" \
  --option-settings
 "Namespace=aws:autoscaling:launchconfiguration,OptionName=ImageId,Value=
${copied_ami_id}" \
  --region "$REGION" \
  --query "Status" \
  --output text)
echo "Environment ${ENVIRONMENT_NAME} is now ${environment_status}"

echo "Waiting for environment ${ENVIRONMENT_NAME} update to complete"
aws elasticbeanstalk wait environment-updated \
  --application-name "$APPLICATION_NAME" \
  --environment-names "$ENVIRONMENT_NAME" \
  --region "$REGION"
echo "Environment ${ENVIRONMENT_NAME} update complete"
```

> **Note**
>
> 必须安装 AWS CLI 才能执行脚本。有关安装说明，请参见《AWS Command Line Interface 用户指南》中的安装或更新最新版本的 AWS CLI。
> 安装 AWS CLI 后，还必须将其配置为使用拥有该环境的 AWS 账户。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI。该账户还必须具有创建 AMI 和更新 Elastic Beanstalk 环境的权限。

这些步骤描述了脚本遵循的过程。

1. 打印正在使用的账户。
2. 确定环境使用的 AMI（源 AMI）。
3. 检查源 AMI 是否已由账户拥有。如果是，请退出。
4. 确定源 AMI 的名称，以便在新 AMI 名称中使用。这也用于确认对源 AMI 的访问。

5. 检查源 AMI 是否已复制到账户。方法是，搜索具有账户所拥有的复制 AMI 名称的 AMI。如果在脚本执行之间更改了 AMI 名称，则需再次复制映像。
6. 如果尚未复制源 AMI，请将源 AMI 复制到账户，然后等待新的 AMI 可用。
7. 更新环境配置以使用新的 AMI。
8. 等待环境更新完成。

从 copy_ami_and_update_env.zip 文件中提取脚本后，执行以下示例运行脚本。将示例中的应用程序名称和环境名称替换为您自己的值。

```
>sh copy_ami_and_update_env.sh \
  --application-name my-application \
  --environment-name my-environment \
  --region us-east-1
```

> **Note**
>
> 为了最大限度地降低存储成本，当您不再需要它来启动 Elastic Beanstalk 环境时，可以考虑清理您的自定义 AMI。有关更多信息，请参阅清除自定义 AMI。

提供静态文件

为了提高性能，您可以配置代理服务器，从 Web 应用程序内的一组目录中提供静态文件（例如 HTML 或图像）。当代理服务器收到对指定路径下的某个文件的请求时，它将直接提供此文件，而不是将请求路由至您的应用程序。

Elastic Beanstalk 支持将代理配置为在基于 Amazon Linux 2 的大多数平台分支上提供静态文件。唯一的例外是 Docker。

> **Note**
>
> 默认情况下，在 Python 和 Ruby 平台上，Elastic Beanstalk 配置一些静态文件夹。有关详细信息，请参阅 Python 和 Ruby 的静态文件配置部分。您可以按照本页中的说明配置其他文件夹。

使用控制台配置静态文件

配置代理服务器提供静态文件

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 滚动到 Platform software（平台软件）部分并找到 Static files（静态文件）组。

   a. 要添加静态文件映射，请选择 Add static files（添加静态文件）。在出现的额外一行中，您将输入提供静态文件的路径和包含要提供的静态文件的目录。

   - 在 Path（路径）字段中，路径名称以斜杠 (/) 开头（例如，“/images”）。
   - 在 Directory（目录）字段中，指定位于应用程序源代码根目录中的目录名称。不要用斜杠开头（例如，“static/image-files”）。

   > **Note**
   >
   > 如果看不到静态文件部分，您必须使用配置文件至少添加一个映射。有关详细信息，请参阅此页上的 the section called “使用配置选项配置静态文件”。

   b. 要删除映射，请选择 Remove（删除）。
6. 要保存更改，请选择页面底部的 Apply（应用）。

使用配置选项配置静态文件

您可以使用配置文件来配置静态文件路径，以及使用配置选项来配置目录位置。您可以将配置文件添加到应用程序的源包中，并在创建环境时或稍后的部署期间进行部署。

如果您的环境使用基于 Amazon Linux 2 的平台分支，请使用 `aws:elasticbeanstalk:environment:proxy:staticfiles` 命名空间。

以下示例配置文件告诉代理服务器在 `/html` 路径的文件夹 `statichtml` 中提供文件，并在路径 `/images` 的 `staticimages` 文件夹中提供文件。

Example .ebextensions/static-files.config

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /html: statichtml
    /images: staticimages
```

如果您的 Elastic Beanstalk 环境使用 Amazon Linux AMI 平台版本（在 Amazon Linux 2 之前），请阅读以下附加信息：

Amazon Linux AMI 平台特定的命名空间

在 Amazon Linux AMI 平台分支上，静态文件配置命名空间因平台而异。如需了解详情，请参考以下页面之一：

- Go 配置命名空间
- Java SE 配置命名空间
- Tomcat 配置命名空间
- Node.js 配置命名空间
- Python 配置命名空间

为 Elastic Beanstalk 环境配置 HTTPS

如果您已为 Elastic Beanstalk 环境购买并配置了自定义域名，您可以使用 HTTPS 来允许用户安全连接到网站。如果您没有域名，仍可将包含自签名证书的 HTTPS 用于开发和测试目的。HTTPS 是传输用户数据或登录信息的任何应用程序而言都必不可少。

配合 Elastic Beanstalk 环境使用 HTTPS 的最简单方式是向环境的负载均衡器分配服务器证书。当您配置负载均衡器以终止 HTTPS 时，客户端和负载均衡器之间的连接是安全的。负载均衡器和 EC2 实例的后端连接使用 HTTP，因此不需要实例的其他配置。

> **Note**
>
> 使用 AWS Certificate Manager (ACM)，您可为自己的域名免费创建可信证书。ACM 证书只能用于 AWS 负载均衡器和 Amazon CloudFront 分配，ACM 仅在特定 AWS 区域中可用。要将 ACM 证书与 Elastic Beanstalk 结合使用，请参阅配置 Elastic Beanstalk 环境的负载均衡器以终止 HTTPS。

如果您在单实例环境中运行应用程序，或需要确保可一直连接到负载均衡器背后的 EC2 实例，您可配置在实例上运行的代理服务器以终止 HTTPS。配置实例以终止 HTTPS 连接要求使用配置文件来更改在实例上运行的软件，并修改安全组，以允许安全连接。

对于负载均衡环境中的端对端 HTTPS，您可以组合实例和负载均衡器终止，以同时加密两个连接。默认情况下，如果您使用 HTTPS 配置负载均衡器以转发流量，它将信任由后端实例向其提交的所有证书。为了实现最高安全性，您将策略与负载均衡器连接，避免其连接未提交其信任的公有证书的实例。

> **Note**
>
> 您也可将负载均衡器配置为传递 HTTPS 流量，无需解密。这种方法的缺点在于负载均衡器无法查看请求，因此无法优化路由或报告响应指标。

如果 ACM 在您所在的区域不可用，您可从第三方购买的可信证书。第三方证书可用于在负载均衡器、后端实例或负载均衡器及后端实例上解密 HTTPS 流量。

对于开发和测试，您可使用开源工具自行创建和签署证书。自签名证书可免费轻松创建，但无法用于公共站点的前端解密。如果您试图通过客户端为 HTTPS 连接使用自签名证书，用户的浏览器将会显示错误消息，表示网站不安全。但您可使用自签名证书来确保后端连接，不会出现问题。

ACM 是以编程方式预置、管理和部署您的服务器证书的首选工具，也可以使用 AWS CLI。如果 ACM 在您所在的 AWS 区域不可用，您可以使用 AWS CLI 上载第三方或自签名证书和私有密钥到 AWS Identity and Access Management (IAM)。在 IAM 中存储的证书可配合负载均衡器和 CloudFront 分配使用。

> **Note**
>
> GitHub 上的是否有 Snake？示例应用程序包括使用 Tomcat Web 应用程序配置 HTTPS 的每种方法的配置文件和说明。请参阅 readme 文件和 HTTPS 说明以了解详细信息。

主题

创建和签名 X509 证书

您可以使用 OpenSSL 为应用程序创建 X509 证书。OpenSSL 一种支持广泛加密功能的标准开源库，其功能包括创建和签名 X509 证书。有关 OpenSSL 的更多信息，请访问 www.openssl.org/。

> **Note**
>
> 如果要在单实例环境中使用 HTTPS 或使用自签名证书在后端上重新加密，您只需在本地创建证书。如果您拥有域名，则可以在 AWS 中创建证书并通过 AWS Certificate Manager (ACM) 免费将证书用于负载均衡的环境。有关说明，请参阅 AWS Certificate Manager 用户指南中的请求证书。

在命令行运行 `openssl version`，查看是否已安装 OpenSSL。如果尚未安装，您可使用公有 GitHub 存储库的说明或使用您喜欢的程序包管理器，构建并安装源代码。OpenSSL 也安装在 Elastic Beanstalk 的 Linux 镜像上，因此另一个快捷的替代方案是使用 EB CLI 的 eb ssh 命令，在正在运行的环境中连接至 EC2 实例：

```
~/eb$ eb ssh
[ec2-user@ip-255-55-55-255 ~]$ openssl version
OpenSSL 1.0.1k-fips 8 Jan 2015
```

您需要创建 RSA 私有密钥才能创建证书签名请求 (CSR)。要创建私有密钥，请使用 openssl genrsa 命令：

```
[ec2-user@ip-255-55-55-255 ~]$ openssl genrsa 2048 > privatekey.pem
```

```
Generating RSA private key, 2048 bit long modulus
.....................................................................................................
+++
..............+++
e is 65537 (0x10001)
```

privatekey.pem

要用于保存私有密钥的文件的名称。通常，openssl genrsa 命令将私有密钥内容输出到屏幕，但此命令将输出通过管道传送到文件。选择任何文件名，并将文件存储在安全位置，以便您以后可以检索它。如果您丢失私有密钥，则无法使用您的证书。

CSR 是指您为申请数字服务器证书而发送至证书颁发机构 (CA) 的文件。要创建 CSR，请使用 openssl req 命令：

```
$ openssl req -new -key privatekey.pem -out csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

输入所请求的信息并按 Enter。下表描述并显示每个字段的示例。

名称	描述	示例
国家/地区名称	代表国家/地区的两个字母 ISO 缩写。	US = 美国
州或省	组织所在州或省的名称。不能缩写此名称。	Washington
所在地名称	组织所在城市的名称。	Seattle
组织名称	组织的法定全称。请勿缩写组织名称。	Example Corporation
组织部门	可选，用于提供额外的组织信息。	市场营销
公用名	网站的完全限定域名。这必须与用户访问站点时看到的域名相符，否则将显示证书错误。	www.example.com

名称	描述	示例
电子邮件地址	站点管理员的电子邮件地址。	someone@example.com

您可向第三方提交签名请求以供签名，或自行签名以供开发和测试。自签名证书也在负载均衡器和 EC2 实例之间用于后端 HTTPS。

要对证书进行签名，请使用 openssl x509 命令。以下示例使用来自上一步骤 (*privatekey.pem*) 和签名请求 (*csr.pem*) 的私有密钥来创建名为 *public.crt* 的公有证书，该证书的有效期为 *365* 天。

```
$ openssl x509 -req -days 365 -in csr.pem -signkey privatekey.pem -out public.crt
Signature ok
subject=/C=us/ST=washington/L=seattle/O=example corporation/OU=marketing/
CN=www.example.com/emailAddress=someone@example.com
Getting Private key
```

保存私钥和公有证书以待将来使用。您可放弃签名请求。请始终将私有密钥存储在安全位置并避免将其添加到源代码。

要将证书用于 Windows Server 平台，您必须将证书转换为 PFX 格式。使用以下命令可在私有密钥和公有证书文件中创建 PFX 证书：

```
$ openssl pkcs12 -export -out example.com.pfx -inkey privatekey.pem -in public.crt
Enter Export Password: password
Verifying - Enter Export Password: password
```

既然您已经拥有了证书，您可以将其上传到 IAM以与负载均衡器结合使用，或者在您的环境中配置实例以终止 HTTPS。

上传证书至 IAM

如需配合 Elastic Beanstalk 环境的负载均衡器使用证书，请将证书和私有密钥上载至 AWS Identity and Access Management (IAM)。您可以将存储在 IAM 中的证书与 Elastic Load Balancing 负载均衡器和 Amazon CloudFront 分配结合使用。

> **Note**
>
> AWS Certificate Manager (ACM) 是预置、管理和部署您的服务器证书的首选工具。有关请求 ACM 证书的更多信息，请参阅 AWS Certificate Manager 用户指南中的请求证书。有关将第三

方证书导入 ACM 中的更多信息，请参阅 AWS Certificate Manager 用户指南中的导入证书。仅当 ACM 在您所在的 AWS 区域不可用时才使用 IAM 上载证书。

您可以使用 AWS Command Line Interface (AWS CLI) 上载证书。以下命令会上传名为 *https-cert.crt* 的带有名为 *private-key.pem* 的私有密钥的自签名证书：

```
$ aws iam upload-server-certificate --server-certificate-name elastic-beanstalk-x509 --
certificate-body file://https-cert.crt --private-key file://private-key.pem
{
    "ServerCertificateMetadata": {
        "ServerCertificateId": "AS5YBEIONO2Q7CAIHKNGC",
        "ServerCertificateName": "elastic-beanstalk-x509",
        "Expiration": "2017-01-31T23:06:22Z",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:server-certificate/elastic-beanstalk-x509",
        "UploadDate": "2016-02-01T23:10:34.167Z"
    }
}
```

`file://` 前缀告知 AWS CLI 加载当前目录中的文件内容。*elastic-beanstalk-x509* 指定该名称调用 IAM 中的证书。

如果您已从证书颁发机构购买了证书并已收到证书链文件，请包括 `--certificate-chain` 选项，将证书链文件同样上传：

```
$ aws iam upload-server-certificate --server-certificate-name elastic-beanstalk-x509 --
certificate-chain file://certificate-chain.pem --certificate-body file://https-cert.crt
 --private-key file://private-key.pem
```

记下证书的 Amazon Resource Name (ARN)。当您更新负载均衡器配置设置以使用 HTTPS 时，您会使用它。

> **Note**
>
> 上传到 IAM 的证书保持存储状态，即使在任何环境的负载均衡器中不再使用它也如此。它包含敏感数据。当您对于任何环境都不再需要该证书时，请务必将其删除。有关从 IAM 中删除证书的详细信息，请参阅https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_server-certs.html#delete-server-certificate。

有关 IAM 中的服务器证书的更多信息，请参阅 IAM 用户指南 中的使用服务器证书。

配置 Elastic Beanstalk 环境的负载均衡器以终止 HTTPS

要更新您的 AWS Elastic Beanstalk 环境以使用 HTTPS，您需要为环境中的负载均衡器配置 HTTPS 侦听器。两种类型的负载均衡器支持 HTTPS 侦听器：经典负载均衡器和 Application Load Balancer。

您可以使用 Elastic Beanstalk 控制台或配置文件配置安全侦听器和分配证书。

> **Note**
>
> 单实例环境没有负载均衡器并且不支持负载均衡器上的 HTTPS 终止。

使用 Elastic Beanstalk 控制台配置安全侦听器

将证书分配至环境的负载均衡器

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Load balancer (负载均衡器) 配置类别中，选择 Edit (编辑)。

   > **Note**
   >
   > 如果 Load balancer (负载均衡器) 配置类别没有 Edit (编辑) 按钮，则表示您的环境没有负载均衡器。

5. 在修改负载均衡器页上，根据与您的环境关联的负载均衡器类型，该过程将会有所不同。
   - 经典负载均衡器
     a. 选择添加侦听器。
     b. 在 经典负载均衡器 listener (经典负载均衡器侦听器) 对话框中，配置以下设置：

- 对于侦听器端口，请键入传入流量端口，通常为 443。
- 对于侦听器协议，选择 HTTPS。
- 对于实例端口，请键入 80。
- 对于实例协议，选择 HTTP。
- 对于 SSL 证书，选择您的证书。

c. 选择 添加。

- 应用程序负载均衡器

a. 选择添加侦听器。

b. 在 Application Load Balancer listener (Application Load Balancer 侦听器) 对话框中，配置以下设置：

- 对于端口，请键入传入流量端口，通常为 443。
- 对于协议，选择 HTTPS。
- 对于 SSL 证书，选择您的证书。

c. 选择 添加。

> **Note**
>
> 对于经典负载均衡器和应用程序负载均衡器，如果下拉菜单中未显示任何证书，您应在 AWS Certificate Manager (ACM) 中为您的自定义域名创建或上传证书（首选）。或者，使用 AWS CLI将证书上传到 IAM。

- Network Load Balancer

a. 选择添加侦听器。

b. 在 Network Load Balancer listener (Network Load Balancer 侦听器) 对话框中，对于 Port (端口)，请键入传入流量端口，通常为 443。

c. 选择 添加。

6. 要保存更改，请选择页面底部的 Apply（应用）。

使用配置文件配置安全侦听器

您可以使用以下配置文件之一在负载均衡器上配置安全侦听器。

Example .ebextensions/securelistener-clb.config

当您的环境具有经典负载均衡器时，请使用此示例。该示例使用 aws:elb:listener 命名空间中的选项通过指定的证书在端口 443 上配置 HTTPS 侦听器，并将解密的流量转发到端口 80 上的环境实例。

```
option_settings:
  aws:elb:listener:443:
    SSLCertificateId: arn:aws:acm:us-east-2:1234567890123:certificate/
####################################
    ListenerProtocol: HTTPS
    InstancePort: 80
```

将突出显示的文本替换为您的证书的 ARN。该证书可以是您在 AWS Certificate Manager (ACM) 中创建或上传的证书（首选），也可以是您使用上传到 IAM 的 AWS CLI证书。

有关经典负载均衡器配置选项的更多信息，请参阅经典负载均衡器配置命名空间。

Example .ebextensions/securelistener-alb.config

当您的环境具有 Application Load Balancer 时，请使用此示例。该示例使用 aws:elbv2:listener 命名空间中的选项通过指定的证书在端口 443 上配置 HTTPS 侦听器。该侦听器将流量路由到默认进程。

```
option_settings:
  aws:elbv2:listener:443:
    ListenerEnabled: 'true'
    Protocol: HTTPS
    SSLCertificateArns: arn:aws:acm:us-east-2:1234567890123:certificate/
####################################
```

Example .ebextensions/securelistener-nlb.config

当您的环境具有 Network Load Balancer 时，请使用此示例。该示例使用 aws:elbv2:listener 命名空间中的选项在端口 443 上配置侦听器。该侦听器将流量路由到默认进程。

```
option_settings:
  aws:elbv2:listener:443:
    ListenerEnabled: 'true'
```

配置安全组

如果将负载均衡器配置为将流量转发到除端口 80 外的实例端口，则必须向安全组添加一个规则来允许来自负载均衡器的入站流量通过该实例端口。如果您在自定义 VPC 中创建环境，则 Elastic Beanstalk 会为您添加此规则。

您可以在 `.ebextensions` 目录的配置文件中为您的应用程序添加 `Resources` 键，从而添加此规则。

以下示例配置文件添加传入规则到 `AWSEBSecurityGroup` 安全组。这样将允许端口 1000 上来自负载均衡器安全组的流量。

Example .ebextensions/sg-ingressfromlb.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 1000
      FromPort: 1000
      SourceSecurityGroupId: {"Fn::GetAtt" : ["AWSEBLoadBalancerSecurityGroup",
 "GroupId"]}
```

配置应用程序以终止实例的 HTTPS 连接

您可使用配置文件配置将流量传递至应用程序的代理服务器，使其终止 HTTPS 连接。如果您希望在单一实例环境中使用 HTTPS，或如果您配置负载均衡器为传递流量而无需解密，这种方式将会十分有用。

要启用 HTTPS，您必须允许在端口 443 向运行 Elastic Beanstalk 应用程序的 EC2 实例传入流量。可以使用配置文件中的 `Resources` 键将针对端口 443 的规则添加到 AWSEBSecurityGroup 安全组的传入规则，从而达到此目的。

以下代码段将传入规则添加到 `AWSEBSecurityGroup` 安全组，该安全组为单一实例环境的所有流量打开端口 443。

`.ebextensions/https-instance-securitygroup.config`

```
Resources:
```

```
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

在默认 Amazon Virtual Private Cloud (Amazon VPC) 的负载均衡环境中，您可以将此策略修改为仅接受来自负载均衡器的流量。有关示例，请参阅在负载均衡的 Elastic Beanstalk 环境中配置端对端加密。

平台

- 在运行 Docker 的 EC2 实例上终止 HTTPS
- 在运行 Go 的 EC2 实例上终止 HTTPS
- 终止运行 Java SE 的 EC2 实例上的 HTTPS
- 在运行 Node.js 的 EC2 实例上终止 HTTPS
- 在运行 PHP 的 EC2 实例上终止 HTTPS
- 在运行 Python 的 EC2 实例上终止 HTTPS
- 在运行 Ruby 的 EC2 实例上终止 HTTPS
- 在运行 Tomcat 的 EC2 实例上终止 HTTPS
- 在运行 .NET Core on Linux 的 Amazon EC2 实例上终止 HTTPS
- 在运行 .NET 的 Amazon EC2 实例上终止 HTTPS

在运行 Docker 的 EC2 实例上终止 HTTPS

对于 Docker 容器，您使用配置文件启用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥资料，并将该文件保存在源包的 `.ebextensions` 目录中。该配置文件执行以下任务：

- `files` 密钥在实例上创建以下文件：

  `/etc/nginx/conf.d/https.conf`

  配置 nginx 服务器。nginx 服务启动时加载此文件。

`/etc/pki/tls/certs/server.crt`

在实例上创建证书文件。将######替换为证书的内容。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

如果您有中间证书，请将其放入 `server.crt` 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

Example .ebextensions/https-instance.config

```
files:
  /etc/nginx/conf.d/https.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      # HTTPS Server

      server {
        listen 443;
        server_name localhost;
```

```
      ssl on;
      ssl_certificate /etc/pki/tls/certs/server.crt;
      ssl_certificate_key /etc/pki/tls/certs/server.key;

      ssl_session_timeout 5m;

      ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
      ssl_prefer_server_ciphers on;

      location / {
        proxy_pass http://docker;
        proxy_http_version 1.1;

        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
      }
    }

/etc/pki/tls/certs/server.crt:
  mode: "000400"
  owner: root
  group: root
  content: |
    -----BEGIN CERTIFICATE-----
    certificate file contents
    -----END CERTIFICATE-----

/etc/pki/tls/certs/server.key:
  mode: "000400"
  owner: root
  group: root
  content: |
    -----BEGIN RSA PRIVATE KEY-----
    private key contents # See note below.
    -----END RSA PRIVATE KEY-----
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 Go 的 EC2 实例上终止 HTTPS

对于 Go 容器类型，您可以使用配置文件和 nginx 配置文件（用于将 nginx 服务器配置为使用 HTTPS）来启用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥占位符，并将该文件保存在源包的 `.ebextensions` 目录中。该配置文件执行以下任务：

- `Resources` 密钥在您的环境实例使用的安全组上启用端口 443。
- `files` 密钥在实例上创建以下文件：

  `/etc/pki/tls/certs/server.crt`

  在实例上创建证书文件。将`######`替换为证书的内容。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

如果您有中间证书，请将其放入 `server.crt` 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

- `container_commands` 密钥在一切配置完成后重启 nginx 服务器，以便让该服务器加载 nginx 配置文件。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----
```

```
container_commands:
  01restart_nginx:
    command: "service nginx restart"
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

将以下内容放置在源包的 .conf 目录内扩展名为 .ebextensions/nginx/conf.d/ 的文件中 (例如 .ebextensions/nginx/conf.d/https.conf)。将 *app_port* 替换为您的应用程序侦听的端口号。此示例配置 nginx 服务器使用 SSL 侦听端口 443。有关 Go 平台上的这些配置文件的更多信息，请参阅 配置反向代理。

Example .ebextensions/nginx/conf.d/https.conf

```
# HTTPS server

server {
    listen       443;
    server_name  localhost;

    ssl                  on;
    ssl_certificate      /etc/pki/tls/certs/server.crt;
    ssl_certificate_key  /etc/pki/tls/certs/server.key;

    ssl_session_timeout  5m;

    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers   on;

    location / {
        proxy_pass  http://localhost:app_port;
        proxy_set_header   Connection "";
        proxy_http_version 1.1;
        proxy_set_header        Host            $host;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
```

```
    }
}
```

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

终止运行 Java SE 的 EC2 实例上的 HTTPS

对于 Java SE 容器类型，您可使用 .ebextensions 配置文件和 nginx 配置文件（用于将 nginx 服务器配置为使用 HTTPS）来启用 HTTPS。

所有 AL2023/AL2 平台都支持统一的代理配置功能。有关在运行 AL2023/AL2 的平台版本上配置代理服务器的更多信息，请展开 the section called “扩展 Linux 平台” 中的反向代理配置部分。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥占位符，并将该文件保存在 `.ebextensions` 目录中。该配置文件执行以下任务：

- `files` 密钥在实例上创建以下文件：

  `/etc/pki/tls/certs/server.crt`

  在实例上创建证书文件。将`######`替换为证书的内容。

  > **Note**
  >
  > YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

如果您有中间证书，请将其放入 server.crt 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

/etc/pki/tls/certs/server.key

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

- container_commands 密钥在一切配置完成后重启 nginx 服务器，以便让该服务器加载 nginx 配置文件。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "service nginx restart"
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅 在 Amazon S3 中安全地存储私有密钥。

将以下内容放置在源包的 `.conf` 目录内扩展名为 `.ebextensions/nginx/conf.d/` 的文件中 (例如 `.ebextensions/nginx/conf.d/https.conf`)。将 *app_port* 替换为您的应用程序侦听的端口号。此示例配置 nginx 服务器使用 SSL 侦听端口 443。有关 Java SE 平台上的这些配置文件的更多信息，请参阅 配置反向代理。

Example .ebextensions/nginx/conf.d/https.conf

```
# HTTPS server

server {
    listen       443;
    server_name  localhost;

    ssl                  on;
    ssl_certificate      /etc/pki/tls/certs/server.crt;
    ssl_certificate_key  /etc/pki/tls/certs/server.key;

    ssl_session_timeout  5m;

    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers   on;

    location / {
        proxy_pass  http://localhost:app_port;
        proxy_set_header   Connection "";
        proxy_http_version 1.1;
        proxy_set_header        Host            $host;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
    }
}
```

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 Node.js 的 EC2 实例上终止 HTTPS

以下示例配置文件将扩展默认 nginx 配置，以在端口 443 上进行侦听，并终止公有证书和私有密钥的 SSL/TLS 连接。

如果已为增强型运行状况报告配置环境，则需要配置 nginx 以生成访问日志。为此，通过删除前导 `# For enhanced health...` 字符，在显示为 # 的注释下方取消对行块的注释。

Example .ebextensions/https-instance.config

```
files:
  /etc/nginx/conf.d/https.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      # HTTPS server

      server {
          listen       443;
          server_name  localhost;

          ssl                  on;
          ssl_certificate      /etc/pki/tls/certs/server.crt;
          ssl_certificate_key  /etc/pki/tls/certs/server.key;
```

```
        ssl_session_timeout  5m;

        ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
        ssl_prefer_server_ciphers   on;

        # For enhanced health reporting support, uncomment this block:

        #if ($time_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {
        #    set $year $1;
        #    set $month $2;
        #    set $day $3;
        #    set $hour $4;
        #}
        #access_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour
healthd;
        #access_log  /var/log/nginx/access.log  main;

        location / {
            proxy_pass  http://nodejs;
            proxy_set_header   Connection "";
            proxy_http_version 1.1;
            proxy_set_header        Host            $host;
            proxy_set_header        X-Real-IP       $remote_addr;
            proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header        X-Forwarded-Proto https;
        }
    }

 /etc/pki/tls/certs/server.crt:
   mode: "000400"
   owner: root
   group: root
   content: |
     -----BEGIN CERTIFICATE-----
     certificate file contents
     -----END CERTIFICATE-----

 /etc/pki/tls/certs/server.key:
   mode: "000400"
   owner: root
   group: root
   content: |
     -----BEGIN RSA PRIVATE KEY-----
```

```
    private key contents # See note below.
    -----END RSA PRIVATE KEY-----
```

files 密钥在实例上创建以下文件：

/etc/nginx/conf.d/https.conf

配置 nginx 服务器。nginx 服务启动时加载此文件。

/etc/pki/tls/certs/server.crt

在实例上创建证书文件。将######替换为证书的内容。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

如果您有中间证书，请将其放入 server.crt 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

/etc/pki/tls/certs/server.key

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 PHP 的 EC2 实例上终止 HTTPS

对于 PHP 容器类型，您可使用配置文件来允许 Apache HTTP Server 使用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥资料，并将该文件保存在源包的 `.ebextensions` 目录中。

该配置文件执行以下任务：

- `packages` 密钥使用 yum 安装 `mod24_ssl`。
- `files` 密钥在实例上创建以下文件：

  `/etc/httpd/conf.d/ssl.conf`

  配置 Apache 服务器。此文件在 Apache 服务启动时加载。

  `/etc/pki/tls/certs/server.crt`

  在实例上创建证书文件。将`######`替换为证书的内容。

  > **Note**
  >
  > YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

如果您有中间证书，请将其放入 server.crt 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

/etc/pki/tls/certs/server.key

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

Example .ebextensions/https-instance.config

```
packages:
  yum:
    mod24_ssl : []

files:
  /etc/httpd/conf.d/ssl.conf:
    mode: "000644"
    owner: root
    group: root
    content: |
      LoadModule ssl_module modules/mod_ssl.so
      Listen 443
      <VirtualHost *:443>
        <Proxy *>
          Order deny,allow
          Allow from all
        </Proxy>

        SSLEngine              on
        SSLCertificateFile     "/etc/pki/tls/certs/server.crt"
        SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"
```

```
        SSLCipherSuite        EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
        SSLProtocol           All -SSLv2 -SSLv3
        SSLHonorCipherOrder   On
        SSLSessionTickets     Off

        Header always set Strict-Transport-Security "max-age=63072000;
includeSubdomains; preload"
        Header always set X-Frame-Options DENY
        Header always set X-Content-Type-Options nosniff

        ProxyPass / http://localhost:80/ retry=0
        ProxyPassReverse / http://localhost:80/
        ProxyPreserveHost on
        RequestHeader set X-Forwarded-Proto "https" early

      </VirtualHost>

 /etc/pki/tls/certs/server.crt:
   mode: "000400"
   owner: root
   group: root
   content: |
     -----BEGIN CERTIFICATE-----
     certificate file contents
     -----END CERTIFICATE-----

 /etc/pki/tls/certs/server.key:
   mode: "000400"
   owner: root
   group: root
   content: |
     -----BEGIN RSA PRIVATE KEY-----
     private key contents # See note below.
     -----END RSA PRIVATE KEY-----
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 Python 的 EC2 实例上终止 HTTPS

对于使用 Apache HTTP Server 和 Web Server Gateway Interface (WSGI) 的 Python 容器类型，您可使用配置文件来允许 Apache HTTP Server 使用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥资料，并将该文件保存在源包的 `.ebextensions` 目录中。该配置文件执行以下任务：

- `packages` 密钥使用 yum 安装 `mod24_ssl`。
- `files` 密钥在实例上创建以下文件：

  `/etc/httpd/conf.d/ssl.conf`

  配置 Apache 服务器。如果您的应用程序的名称不是 `application.py`，请将 `WSGIScriptAlias` 的值中突出显示的文本替换为您的应用程序的本地路径。例如，django 应用程序的路径可能是 `django/wsgi.py`。此位置应与您为环境设置的 `WSGIPath` 选项值相符。

  根据您的应用程序的要求，您可能还需要向 python-path 参数添加其他目录。

  `/etc/pki/tls/certs/server.crt`

  在实例上创建证书文件。将`######`替换为证书的内容。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

如果您有中间证书，请将其放入 `server.crt` 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

- `container_commands` 密钥在一切配置完成后停止 httpd 服务，以便该服务使用新的 `https.conf` 文件和证书。

> **Note**
>
> 该示例仅可用于使用 Python 平台的环境。

Example .ebextensions/https-instance.config

```
packages:
  yum:
    mod24_ssl : []

files:
  /etc/httpd/conf.d/ssl.conf:
```

```
    mode: "000644"
    owner: root
    group: root
    content: |
      LoadModule wsgi_module modules/mod_wsgi.so
      WSGIPythonHome /opt/python/run/baselinenv
      WSGISocketPrefix run/wsgi
      WSGIRestrictEmbedded On
      Listen 443
      <VirtualHost *:443>
        SSLEngine on
        SSLCertificateFile "/etc/pki/tls/certs/server.crt"
        SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"

        Alias /static/ /opt/python/current/app/static/
        <Directory /opt/python/current/app/static>
        Order allow,deny
        Allow from all
        </Directory>

        WSGIScriptAlias / /opt/python/current/app/application.py

        <Directory /opt/python/current/app>
        Require all granted
        </Directory>

        WSGIDaemonProcess wsgi-ssl processes=1 threads=15 display-name=%{GROUP} \
          python-path=/opt/python/current/app \
          python-home=/opt/python/run/venv \
          home=/opt/python/current/app \
          user=wsgi \
          group=wsgi
        WSGIProcessGroup wsgi-ssl

      </VirtualHost>

  /etc/pki/tls/certs/server.crt:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----
```

```
  /etc/pki/tls/certs/server.key:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01killhttpd:
    command: "killall httpd"
  02waitforhttpddeath:
    command: "sleep 3"
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 Ruby 的 EC2 实例上终止 HTTPS

对于 Ruby 容器类型，启用 HTTPS 的方式取决于所用应用程序服务器的类型。

主题

为使用 Puma 的 Ruby 配置 HTTPS

对于使用 Puma 作为应用程序服务器的 Ruby 容器类型，您可使用配置文件启用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥资料，并将该文件保存在源包的 `.ebextensions` 目录中。该配置文件执行以下任务：

- `files` 密钥在实例上创建以下文件：

  `/etc/nginx/conf.d/https.conf`

  配置 nginx 服务器。nginx 服务启动时加载此文件。

  `/etc/pki/tls/certs/server.crt`

  在实例上创建证书文件。将`######`替换为证书的内容。

  > **Note**
  >
  > YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

  如果您有中间证书，请将其放入 `server.crt` 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
```

```
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

- `container_commands` 密钥在一切配置完成后重启 nginx 服务器，以便让该服务器使用新的 `https.conf` 文件。

Example .ebextensions/https-instance.config

```
files:
  /etc/nginx/conf.d/https.conf:
    content: |
      # HTTPS server

      server {
          listen       443;
          server_name  localhost;

          ssl                  on;
          ssl_certificate      /etc/pki/tls/certs/server.crt;
          ssl_certificate_key  /etc/pki/tls/certs/server.key;

          ssl_session_timeout  5m;

          ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
          ssl_prefer_server_ciphers   on;

          location / {
              proxy_pass  http://my_app;
              proxy_set_header        Host            $host;
              proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
              proxy_set_header        X-Forwarded-Proto https;
          }

          location /assets {
            alias /var/app/current/public/assets;
            gzip_static on;
            gzip on;
            expires max;
            add_header Cache-Control public;
```

```
        }

        location /public {
          alias /var/app/current/public;
          gzip_static on;
          gzip on;
          expires max;
          add_header Cache-Control public;
        }
      }

  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "service nginx restart"
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
```

```
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

为使用 Passenger 的 Ruby 配置 HTTPS

对于使用 Passenger 作为应用程序服务器的 Ruby 容器类型，您可同时使用配置文件和 JSON 文件启用 HTTPS。

为使用 Passenger 的 Ruby 配置 HTTPS

1. 将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥资料，并将该文件保存在源包的 `.ebextensions` 目录中。该配置文件执行以下任务：

   - `files` 密钥在实例上创建以下文件：

     `/etc/pki/tls/certs/server.crt`

     在实例上创建证书文件。将######替换为证书的内容。

     > **Note**
     >
     > YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

     如果您有中间证书，请将其放入 `server.crt` 中您的站点证书的后面：

```
    -----BEGIN CERTIFICATE-----
certificate file contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
first intermediate certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
second intermediate certificate
```

```
-----END CERTIFICATE-----
```

`/etc/pki/tls/certs/server.key`

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

Example 用于为使用 Passenger 的 Ruby 配置 HTTPS 的 .Ebextensions 代码段

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

2. 创建一个文本文件并将以下 JSON 添加到该文件。将该文件保存在源包的名为 `passenger-standalone.json` 的根目录中。此 JSON 文件将 Passenger 配置为使用 HTTPS。

> **Important**
>
> 此 JSON 文件不得包含字节顺序标记 (BOM)。如果此文件包含字节顺序标记，则 Passenger JSON 库将不会正确读取此文件，并且 Passenger 服务将不会启动。

Example passenger-standalone.json

```
{
  "ssl" : true,
  "ssl_port" : 443,
  "ssl_certificate" : "/etc/pki/tls/certs/server.crt",
  "ssl_certificate_key" : "/etc/pki/tls/certs/server.key"
}
```

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 Tomcat 的 EC2 实例上终止 HTTPS

对于 Tomcat 容器类型，您可使用配置文件来允许 HTTP Server 在充当 Tomcat 的反向代理时使用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥资料，并将该文件保存在源包的 `.ebextensions` 目录中。该配置文件执行以下任务：

- `files` 密钥在实例上创建以下文件：

  `/etc/pki/tls/certs/server.crt`

  在实例上创建证书文件。将`######`替换为证书的内容。

> **Note**
>
> YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

`/etc/pki/tls/certs/server.key`

在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

`/opt/elasticbeanstalk/hooks/appdeploy/post/99_start_httpd.sh`

创建部署后挂钩脚本以重新启动 httpd 服务。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    mode: "000400"
    owner: root
    group: root
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

  /opt/elasticbeanstalk/hooks/appdeploy/post/99_start_httpd.sh:
    mode: "000755"
    owner: root
    group: root
    content: |
```

```
      #!/usr/bin/env bash
      sudo service httpd restart
```

您还必须将环境的代理服务器配置为在端口 443 上进行侦听。以下 Apache 2.4 配置将在端口 443 上添加一个侦听器。要了解更多信息，请参阅配置 Tomcat 环境的代理服务器。

Example .ebextensions/httpd/conf.d/ssl.conf

```
Listen 443
<VirtualHost *:443>
  ServerName server-name
  SSLEngine on
  SSLCertificateFile "/etc/pki/tls/certs/server.crt"
  SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"

  <Proxy *>
    Require all granted
  </Proxy>
  ProxyPass / http://localhost:8080/ retry=0
  ProxyPassReverse / http://localhost:8080/
  ProxyPreserveHost on

  ErrorLog /var/log/httpd/elasticbeanstalk-ssl-error_log

</VirtualHost>
```

您的证书供应商可能包含中间证书，可安装此证书以提高与移动客户端的兼容性。使用中间证书颁发机构 (CA) 捆绑配置 Apache，方法是将以下内容添加到您的 SSL 配置文件 (如需了解位置，请参阅扩展并覆盖默认 Apache 配置 — Amazon Linux AMI (AL1))：

- 在 ssl.conf 文件内容中，指定链文件：

```
SSLCertificateKeyFile "/etc/pki/tls/certs/server.key"
SSLCertificateChainFile "/etc/pki/tls/certs/gd_bundle.crt"
SSLCipherSuite        EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
```

- 使用中间证书的内容向 files 密钥添加新条目：

```
files:
  /etc/pki/tls/certs/gd_bundle.crt:
    mode: "000400"
```

```
    owner: root
    group: root
    content: |
      -----BEGIN CERTIFICATE-----
      First intermediate certificate
      -----END CERTIFICATE-----
      -----BEGIN CERTIFICATE-----
      Second intermediate certificate
      -----END CERTIFICATE-----
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 .NET Core on Linux 的 Amazon EC2 实例上终止 HTTPS

对于 .NET Core on Linux 容器类型，您可以使用 `.ebextensions` 配置文件和 nginx 配置文件（用于配置 nginx 服务器以使用 HTTPS）来启用 HTTPS。

将以下代码段添加到您的配置文件中，按照提示替换证书和私有密钥占位符，并将该文件保存在 .ebextensions 目录中。该配置文件执行以下任务：

- files 密钥在实例上创建以下文件：

  /etc/pki/tls/certs/server.crt

  在实例上创建证书文件。将######替换为证书的内容。

  > **Note**
  >
  > YAML 依赖一致的缩进。当替换示例配置文件中的内容时，应匹配缩进级别，并且确保您的文本编辑器使用空格而不是字符来进行缩进。

  如果您有中间证书，请将其放入 server.crt 中您的站点证书的后面：

  ```
      -----BEGIN CERTIFICATE-----
  certificate file contents
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  first intermediate certificate
  -----END CERTIFICATE-----
  -----BEGIN CERTIFICATE-----
  second intermediate certificate
  -----END CERTIFICATE-----
  ```

  /etc/pki/tls/certs/server.key

  在实例上创建私有密钥文件。将######替换为用于创建证书请求或自签名证书的私有密钥的内容。

- container_commands 密钥在一切配置完成后重启 nginx 服务器，以便让该服务器加载 nginx 配置文件。

Example .ebextensions/https-instance.config

```
files:
  /etc/pki/tls/certs/server.crt:
    content: |
      -----BEGIN CERTIFICATE-----
```

```
      certificate file contents
      -----END CERTIFICATE-----

  /etc/pki/tls/certs/server.key:
    content: |
      -----BEGIN RSA PRIVATE KEY-----
      private key contents # See note below.
      -----END RSA PRIVATE KEY-----

container_commands:
  01restart_nginx:
    command: "systemctl restart nginx"
```

> **Note**
>
> 避免将包含私有密钥的配置文件提交到源控件。在测试完配置并确认其有效后，请在 Amazon S3 中存储私有密钥并修改配置以在部署期间下载该密钥。有关说明，请参阅在 Amazon S3 中安全地存储私有密钥。

将以下内容放置在源代码包的 `.conf` 目录内扩展名为 `.platform/nginx/conf.d/` 的文件中（例如 `.platform/nginx/conf.d/https.conf`）。将 *`app_port`* 替换为您的应用程序侦听的端口号。此示例配置 nginx 服务器使用 SSL 侦听端口 443。有关 .NET Core on Linux 平台上的这些配置文件的更多信息，请参阅the section called “代理服务器”。

Example .platform/nginx/conf.d/https.conf

```
# HTTPS server

server {
    listen       443 ssl;
    server_name  localhost;

    ssl_certificate      /etc/pki/tls/certs/server.crt;
    ssl_certificate_key  /etc/pki/tls/certs/server.key;

    ssl_session_timeout  5m;

    ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers   on;
```

```
    location / {
        proxy_pass  http://localhost:app_port;
        proxy_set_header   Connection "";
        proxy_http_version 1.1;
        proxy_set_header        Host            $host;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto https;
    }
}
```

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID，并向其中添加规则。

Example .ebextensions/https-instance-single.config

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

对于负载均衡环境，将负载均衡器配置为保持不变地传输安全流量，或者解密后重新加密，以便实现端到端加密。

在运行 .NET 的 Amazon EC2 实例上终止 HTTPS

以下配置文件创建并运行执行以下任务的 Windows PowerShell 脚本：

- 检查是否存在绑定到端口 443 的现有 HTTPS 证书。
- 从 Amazon S3 存储桶获取 PFX 证书。

> **Note**
>
> 向中添加访问亚马逊 S3 存储桶中的 SSL 证书的`AmazonS3ReadOnlyAccess`策略。`aws-elasticbeanstalk-ec2-role`

- 从中获取密码 AWS Secrets Manager。

> **Note**
>
> 在中aws-elasticbeanstalk-ec2-role添加一条允许对包含证书密码的密钥secretsmanager:GetSecretValue执行操作的语句

- 安装证书。
- 将证书绑定到端口 443。

> **Note**
>
> 要删除 HTTP 终端节点（端口 80），请在示例的删除 HTTP 绑定 部分中包括 Remove-WebBinding 命令。

Example .ebextensions/ .config https-instance-dotnet

```
files:
  "C:\\certs\\install-cert.ps1":
    content: |
      import-module webadministration
      ## Settings - replace the following values with your own
      $bucket = "DOC-EXAMPLE-BUCKET"  ## S3 bucket name
      $certkey = "example.com.pfx"    ## S3 object key for your PFX certificate
      $secretname = "example_secret"  ## AWS Secrets Manager name for a secret that
 contains the certificate's password
      ##

      # Set variables
      $certfile = "C:\cert.pfx"
      $pwd = Get-SECSecretValue -SecretId $secretname | select -expand SecretString

      # Clean up existing binding
      if ( Get-WebBinding "Default Web Site" -Port 443 ) {
        Echo "Removing WebBinding"
        Remove-WebBinding -Name "Default Web Site" -BindingInformation *:443:
      }
      if ( Get-Item -path IIS:\SslBindings\0.0.0.0!443 ) {
        Echo "Deregistering WebBinding from IIS"
        Remove-Item -path IIS:\SslBindings\0.0.0.0!443
```

```
      }

      # Download certificate from S3
      Read-S3Object -BucketName $bucket -Key $certkey -File $certfile

      # Install certificate
      Echo "Installing cert..."
      $securepwd = ConvertTo-SecureString -String $pwd -Force -AsPlainText
      $cert = Import-PfxCertificate -FilePath $certfile cert:\localMachine\my -Password
 $securepwd

      # Create site binding
      Echo "Creating and registering WebBinding"
      New-WebBinding -Name "Default Web Site" -IP "*" -Port 443 -Protocol https
      New-Item -path IIS:\SslBindings\0.0.0.0!443 -value $cert -Force

      ## Remove the HTTP binding
      ## (optional) Uncomment the following line to unbind port 80
      # Remove-WebBinding -Name "Default Web Site" -BindingInformation *:80:
      ##

      # Update firewall
      netsh advfirewall firewall add rule name="Open port 443" protocol=TCP
 localport=443 action=allow dir=OUT

commands:
  00_install_ssl:
    command: powershell -NoProfile -ExecutionPolicy Bypass -file C:\\certs\\install-
cert.ps1
```

在单实例环境中，您还必须修改实例的安全组，以便允许端口 443 上的流量。以下配置文件使用 AWS CloudFormation 函数检索安全组的 ID 并向其添加规则。

Example .ebextensions/ .config https-instance-single

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
```

```
      CidrIp: 0.0.0.0/0
```

对于负载平衡的环境，您可以将负载均衡器配置为要么原封不动地传递安全流量，要么解密并重新加密以进行加密。 end-to-end

在负载均衡的 Elastic Beanstalk 环境中配置端对端加密

终止负载均衡器的安全连接并在后端上使用 HTTP 可能对应用程序已经足够。即使在同一账户下运行，AWS 资源之间的网络流量也不会被不属于连接的实例侦听。

但是，如果您正在开发需要遵守严格外部规定的应用程序，则可能需要保护所有网络连接。您可以使用 Elastic Beanstalk 控制台或配置文件，将 Elastic Beanstalk 环境的负载均衡器安全地连接到后端实例以满足这些要求。以下过程将重点放在配置文件上。

首先，在负载均衡器中添加一个安全侦听器（如果尚未添加）。

您还必须配置环境中的实例，以侦听安全端口并终止 HTTPS 连接。此配置根据不同平台而有所变化。有关说明，请参阅配置应用程序以终止实例的 HTTPS 连接。您可为 EC2 实例使用自签名证书，不会出现问题。

接下来，配置侦听器在应用程序使用的端口上使用 HTTPS 转发流量。根据您的环境使用的负载均衡器类型，使用以下配置文件之一。

`.ebextensions/https-reencrypt-clb.config`

将此配置文件与 Classic Load Balancer 配合使用。除了配置负载均衡器以外，该配置文件还更改默认运行状况检查以使用端口 443 和 HTTPS，以确保负载均衡器可以安全地进行连接。

```
option_settings:
  aws:elb:listener:443:
    InstancePort: 443
    InstanceProtocol: HTTPS
  aws:elasticbeanstalk:application:
    Application Healthcheck URL: HTTPS:443/
```

`.ebextensions/https-reencrypt-alb.config`

将此配置文件与 Application Load Balancer 配合使用。

```
option_settings:
```

```
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
    Protocol: HTTPS
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
    Protocol: HTTPS
```

.ebextensions/https-reencrypt-nlb.config

将此配置文件与 Network Load Balancer 配合使用。

```
option_settings:
  aws:elbv2:listener:443:
    DefaultProcess: https
    ListenerEnabled: 'true'
  aws:elasticbeanstalk:environment:process:https:
    Port: '443'
```

`DefaultProcess` 选项以此方式命名是因为 Application Load Balancer，它可能在同一端口上具有非默认侦听器，以用于侦听到特定路径的流量（如需详细信息，请参阅 应用程序负载均衡器）。对于 Network Load Balancer，该选项为该侦听器指定唯一的目标进程。

在此示例中，我们将进程命名为 `https`，因为它侦听安全 (HTTPS) 流量。由于 Network Load Balancer 只能使用 TCP，因此该侦听器使用 TCP 协议将流量发送到指定端口上的进程。由于 HTTP 和 HTTPS 网络流量是在 TCP 上实施的，因此，可以这样做。

> **Note**
>
> EB CLI 和 Elastic Beanstalk 控制台会对前面的选项应用建议的值。如果您需要使用配置文件来配置相同的项，则必须删除这些设置。有关更多信息，请参阅 建议值。

在下一个任务中，您需要将负载均衡器的安全组修改为允许流量。根据您启动环境所在的 Amazon Virtual Private Cloud (Amazon VPC)（默认 VPC 或自定义 VPC），负载均衡器的安全组将有所不同。在默认 VPC 中，Elastic Load Balancing 提供可供所有负载均衡器使用的默认安全组。在您创建的 Amazon VPC 中，Elastic Beanstalk 会创建一个安全组供负载均衡器使用。

为了同时支持两种方案，您可创建一个安全组并告知 Elastic Beanstalk 使用该安全组。以下配置文件创建安全组并将其连接到负载均衡器。

.ebextensions/https-lbsecuritygroup.config

```
option_settings:
  # Use the custom security group for the load balancer
  aws:elb:loadbalancer:
    SecurityGroups: '`{ "Ref" : "loadbalancersg" }`'
    ManagedSecurityGroup: '`{ "Ref" : "loadbalancersg" }`'

Resources:
  loadbalancersg:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: load balancer security group
      VpcId: vpc-########
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
```

用默认或自定义 VPC ID 替换突出显示的文本。上述示例包括通过端口 80 的入口和出口，以允许 HTTP 连接。如果您只想要允许安全连接，则可删除这些属性。

最后，添加允许在负载均衡器的安全组和实例的安全组之间通过端口 443 进行通信的入口和出口规则。

.ebextensions/https-backendsecurity.config

```
Resources:
  # Add 443-inbound to instance security group (AWSEBSecurityGroup)
  httpsFromLoadBalancerSG:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
```

```
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      SourceSecurityGroupId: {"Fn::GetAtt" : ["loadbalancersg", "GroupId"]}
  # Add 443-outbound to load balancer security group (loadbalancersg)
  httpsToBackendInstances:
    Type: AWS::EC2::SecurityGroupEgress
    Properties:
      GroupId: {"Fn::GetAtt" : ["loadbalancersg", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      DestinationSecurityGroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
```

与创建安全组分离执行这一步骤允许您限制源位置和目标位置安全组，而无需创建循环依赖项。

完成以上所有任务后，负载均衡器可使用 HTTPS 安全地连接至后端实例。负载均衡器不关心实例证书为自签名还有由可信赖的证书颁发机构颁发，并将接受向它提交的任何证书。

您可为负载均衡器添加策略，告知它只可信任某个特定证书，从而更改此行为。以下配置文件创建两个策略。一个策略指定公有证书，另一个策略则告知负载均衡器只可为实例端口 443 的连接信任该证书。

.ebextensions/https-backendauth.config

```
option_settings:
  # Backend Encryption Policy
  aws:elb:policies:backendencryption:
    PublicKeyPolicyNames: backendkey
    InstancePorts:  443
  # Public Key Policy
  aws:elb:policies:backendkey:
    PublicKey: |
      -----BEGIN CERTIFICATE-----
      ################################################################
      ################################################################
      ################################################################
      ################################################################
      ################################################
      -----END CERTIFICATE-----
```

用 EC2 实例的公有证书内容替换突出显示的文本。

为 TCP 传递配置环境的负载均衡器

如果您不希望 AWS Elastic Beanstalk 环境中的负载均衡器对 HTTPS 流量进行解码，可配置安全侦听器将请求原样转发至后端实例。

首先将环境的 EC2 实例配置为终止 HTTPS。在为组合添加负载均衡器之前，在单一实例环境中测试配置，以确保一切正常运行。

将配置文件添加到您的项目，以便在端口 443 上配置监听器，该端口将 TCP 包原样传递至后端实例上的端口 443：

.ebextensions/https-lb-passthrough.config

```
option_settings:
  aws:elb:listener:443:
    ListenerProtocol: TCP
    InstancePort: 443
    InstanceProtocol: TCP
```

在默认的 Amazon Virtual Private Cloud (Amazon VPC) 中，您还需要向实例的安全组添加规则，以允许端口 443 上来自负载均衡器的入站流量：

.ebextensions/https-instance-securitygroup.config

```
Resources:
  443inboundfromloadbalancer:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup", "GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      SourceSecurityGroupName: { "Fn::GetAtt": ["AWSEBLoadBalancer",
 "SourceSecurityGroup.GroupName"] }
```

在自定义 VPC 中，Elastic Beanstalk 会为您更新安全组配置。

在 Amazon S3 中安全地存储私有密钥

用于对公有证书签名的私有密钥是私有的，不应提交到源代码。通过将私有密钥上传到 Amazon S3，然后将 Elastic Beanstalk 配置为在应用程序部署期间从 Amazon S3 下载文件，可以避免将私有密钥存储在配置文件中。

以下示例显示了配置文件的资源和文件部分从 Amazon S3 存储桶下载私有密钥文件。

Example .ebextensions/privatekey.config

```
Resources:
  AWSEBAutoScalingGroup:
    Metadata:
      AWS::CloudFormation::Authentication:
        S3Auth:
          type: "s3"
          buckets: ["elasticbeanstalk-us-west-2-123456789012"]
          roleName:
            "Fn::GetOptionSetting":
              Namespace: "aws:autoscaling:launchconfiguration"
              OptionName: "IamInstanceProfile"
              DefaultValue: "aws-elasticbeanstalk-ec2-role"
files:
  # Private key
  "/etc/pki/tls/certs/server.key":
    mode: "000400"
    owner: root
    group: root
    authentication: "S3Auth"
    source: https://elasticbeanstalk-us-west-2-123456789012.s3.us-west-2.amazonaws.com/
server.key
```

将示例中的存储桶名称和 URL 替换为您自己的。此文件中的第一个条目将名为 S3Auth 的身份验证方法添加到环境的 Auto Scaling 组的元数据中。如果您为环境配置了自定义实例配置文件，则将使用自定义实例配置文件，否则应用默认值 aws-elasticbeanstalk-ec2-role。默认实例配置文件具有从 Elastic Beanstalk 存储桶读取的权限。如果您使用不同的存储桶，请向实例配置文件添加权限。

第二个条目使用 S3Auth 身份验证方法从指定 URL 下载私有密钥并将其保存到 /etc/pki/tls/certs/server.key。然后，代理服务器可以从此位置读取私有密钥以终止实例的 HTTPS 连接。

分配到您的环境的 EC2 实例的实例配置文件必须具有从指定存储桶读取密钥对象的权限。验证实例配置文件有权读取 IAM 中的对象，并且存储桶和对象上的权限不禁止实例配置文件。

查看存储桶的权限

1. 打开 Amazon S3 管理控制台。
2. 选择存储桶。
3. 选择属性，然后选择权限。

4. 验证您的账户已被授予存储桶的读取权限。
5. 如果已附加存储桶策略，请选择存储桶策略以查看分配到存储桶的权限。

配置 HTTP 到 HTTPS 重定向

在为 Elastic Beanstalk 环境配置 HTTPS及其子主题中，我们讨论了如何配置您的 Elastic Beanstalk 环境以使用 HTTPS，从而确保应用程序中的流量加密。本主题介绍如何在最终用户仍在启动应用程序时正常处理到该应用程序的 HTTP 流量。为此，您需要配置 HTTP to HTTPS redirection (HTTP 到 HTTPS 重定向)，有时也称为强制 HTTPS。

要配置重定向，您首先将环境配置为处理 HTTPS 流量。然后，您可以将 HTTP 流量重定向到 HTTPS。下面的小节将讨论这两个步骤。

配置您的环境以处理 HTTPS 流量

根据环境的负载均衡配置，执行以下操作之一：

- 负载均衡环境 – 配置负载均衡器以终止 HTTPS。
- 单实例环境 – 配置您的应用程序以终止实例中的 HTTPS 连接。此配置取决于您环境的平台。

将 HTTP 流量重定向到 HTTPS

您可以配置环境实例上的 Web 服务器或环境的 Application Load Balancer，以将 HTTP 流量重定向到 HTTPS。请执行下列操作之一：

- 配置实例 Web 服务器 – 此方法适用于任何 Web 服务器环境。在 Amazon Elastic Compute Cloud (Amazon EC2) 实例上配置 Web 服务器，以使用 HTTP 重定向响应状态来响应 HTTP 流量。此配置取决于您环境的平台。在 GitHub 上的 `https-redirect` 集合中查找您的平台对应的文件夹，然后使用该文件夹中的示例配置文件。

  如果您的环境使用 Elastic Load Balancing 运行状况检查，负载均衡器需要运行状况良好的实例用 HTTP 200 (OK) 响应来响应 HTTP 运行状况检查消息。因此，您的 Web 服务器不应将这些消息重定向到 HTTPS。`https-redirect` 中的示例配置文件可正确处理这一需求。
- 配置负载均衡器 – 如果您的负载均衡环境使用了 Application Load Balancer，则此方法可行。当 HTTP 流量进入时，Application Load Balancer 可以发送重定向响应。在这种情况下，不需要在您环境的实例上配置重定向。我们在 GitHub 上提供了两个示例配置文件，展示了如何配置 Application Load Balancer 进行重定向。`alb-http-to-https-redirection-full.config` 配置文件在端口 443 上创建 HTTPS 侦听器，并修改默认端口 80 侦听器以将传入的 HTTP 流量重定向到

HTTPS。alb-http-to-https-redirection.config 配置文件需要定义 443 侦听器（可以使用标准 Elastic Beanstalk 配置命名空间或 Elastic Beanstalk 控制台）。然后，它负责修改端口 80 侦听器以便执行重定向。

监控环境

当您运行生产网站时，了解您的应用程序是否可用以及能否对请求做出响应非常重要。为了帮助监控应用程序的响应能力，Elastic Beanstalk 提供了一些功能，这些功能可监控有关应用程序的统计数据并创建在超过阈值时触发的警报。

主题

在AWS管理控制台中监控环境运行状况

您可以从 Elastic Beanstalk 控制台中访问有关应用程序的运行信息。控制台简要显示环境的状态和应用程序运行状况。在控制台的 Environments (环境) 页面和每个应用程序的页面中，已对列表上的环境进行了颜色编码来指示状态。

在 Elastic Beanstalk 控制台中监控环境

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Monitoring (监控)。

“监控”页面向您显示有关环境的总体统计数据，如 CPU 使用率和平均延迟。除了总体统计数据之外，您还可以查看随时间推移显示资源使用情况的监控图表。您可以单击任何图表以查看详细信息。

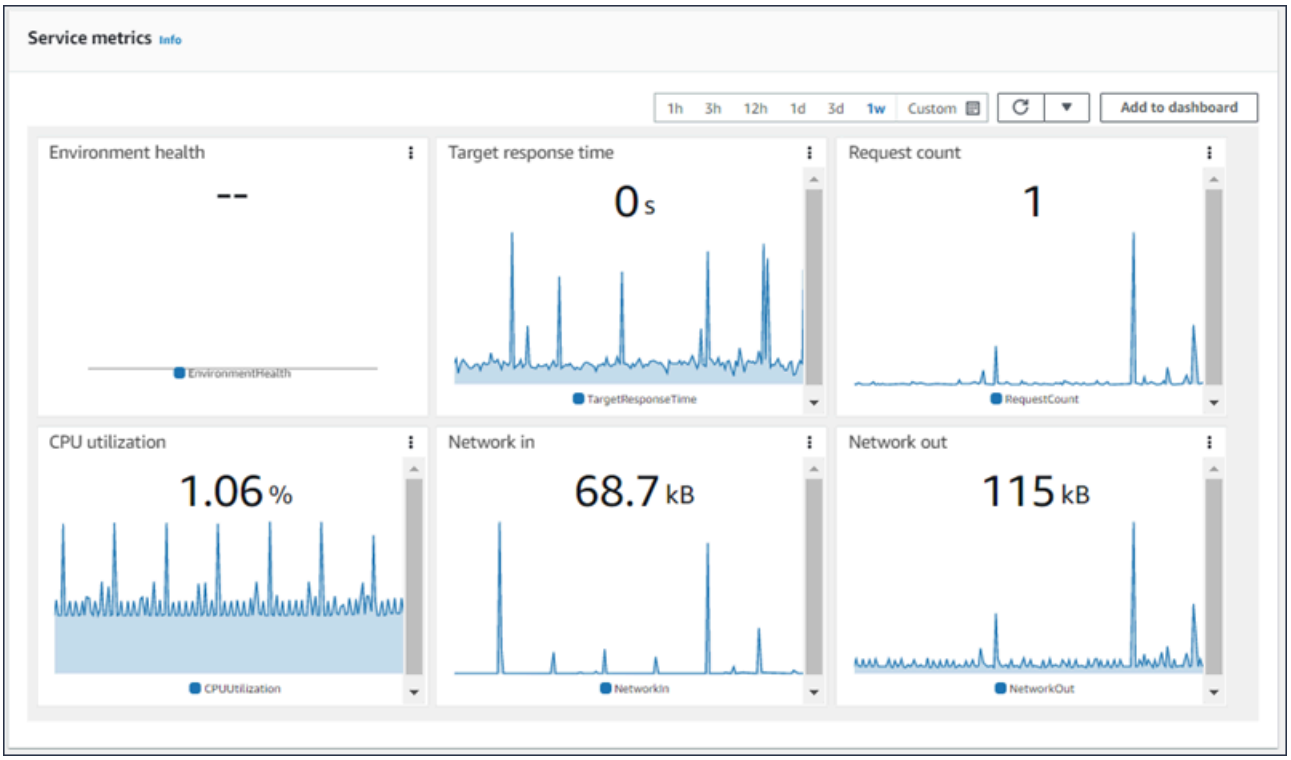
> **Note**
>
> 默认情况下，仅启用基本的 CloudWatch 指标，该指标会在五分钟时间内返回数据。通过编辑环境的配置设置，您可启用更为精确的一分钟 CloudWatch 指标。

监控图表

监控页面显示您的环境的运行状况相关指标概述。这包括 Elastic Load Balancing 和 Amazon EC2 提供的默认指标集，以及显示环境运行状况随时间变化的图表。

图表上方的条形提供了各种时间间隔供您选择。例如，选择 1w 可显示过去一周的信息。或者选择 3h 显示过去三个小时内的信息。

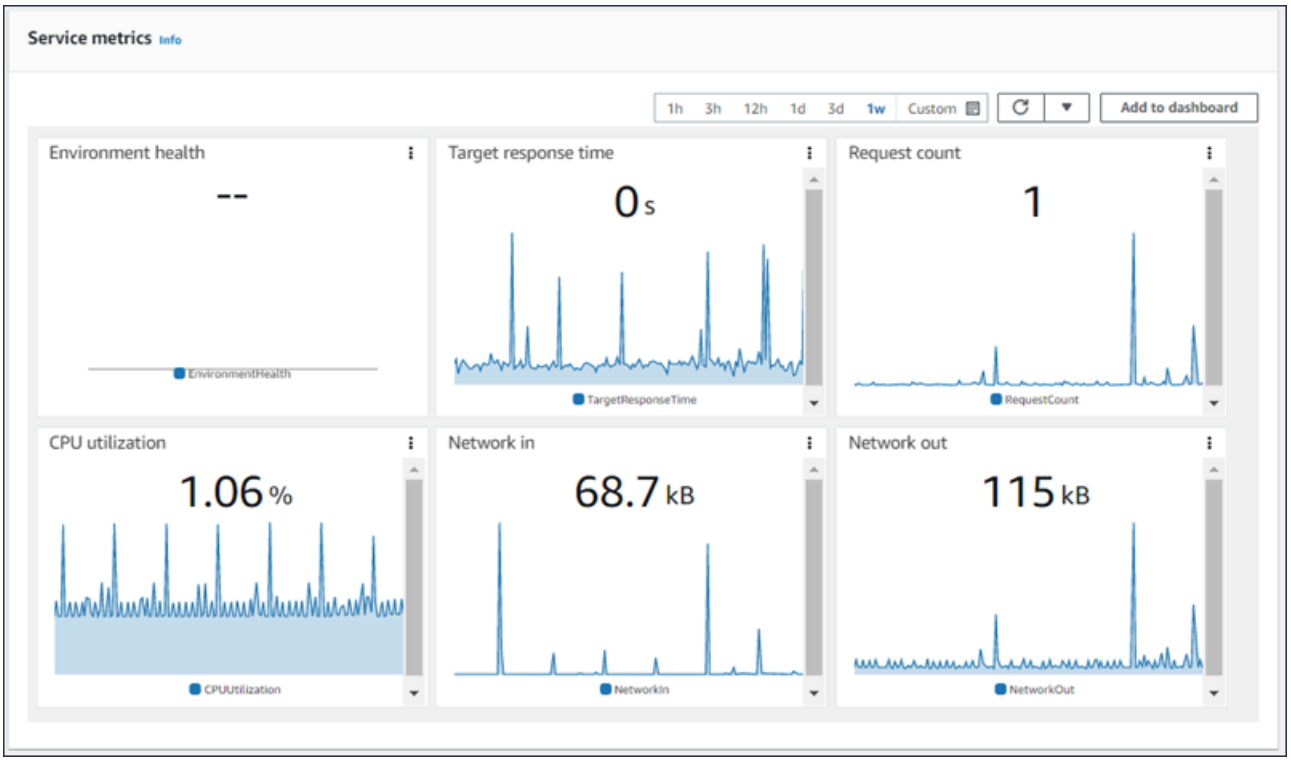
要选择更多种类的时间间隔，请选择自定义。在这里，您有两个范围选项：绝对或相对。绝对选项允许您指定特定的日期范围，例如 2023 年 1 月 1 日到 2023 年 6 月 30 日。相对选项允许选择具有特定时间单位的整数：分钟、小时、天、周或月。示例包括 10 小时、10 天和 10 个月。



自定义监控控制台

要创建和查看自定义指标，您必须使用 Amazon CloudWatch。使用 CloudWatch，您可以创建自定义控制面板以在单一视图中监控资源。选择添加到控制面板，从监控页面导航到 Amazon CloudWatch 控

制台。Amazon CloudWatch 为您提供了创建新控制面板或选择现有控制面板的选项。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用 Amazon CloudWatch 控制面板。



为所有环境启用 Elastic Load Balancing 和 Amazon EC2 指标。

使用 增强型运行状况 时，EnvironmentHealth 指标将会启用，并自动向监控控制台添加一个图表。增强型运行状况还可将运行状况页面添加到管理控制台。有关可用的增强型运行状况指标的列表，请参阅发布环境的 Amazon CloudWatch 自定义指标。

基本运行状况报告

AWS Elastic Beanstalk 使用来自多个来源的信息来确定您的环境是否可用并处理来自 Internet 的请求。环境的运行状况由四种颜色之一表示，并显示在 Elastic Beanstalk 控制台的环境概述页面上。它也可以DescribeEnvironments通过 API 获得，也可以通过 EB CLI 调用eb status。

在版本 2 Linux 平台版本之前，唯一的运行状况报告系统为基本运行状况。基本运行状况报告系统根据 Elastic Load Balancing 为负载均衡的环境或 Amazon Elastic Compute Cloud 为单实例环境执行的运行状况检查，提供有关 Elastic Beanstalk 环境中的实例运行状况的信息。

除了检查 EC2 实例的运行状况外，Elastic Beanstalk 还会监控您环境中的其他资源，并报告可能会导致环境对用户不可用的缺失或错误配置的资源。

您的环境中的资源收集的指标每隔五分钟就会发布到 Amazon CloudWatch 。这包括来自 EC2 的操作系统指标以及来自 Elastic Load Balancing 的请求指标。您可以在环境控制台的 “监控” 页面上查看基于这些 CloudWatch 指标的图表。对于基本运行状况，这些指标将不会用于确定环境的运行状况。

主题

运行状况颜色

Elastic Beanstalk 根据 Web 服务器环境中运行的应用程序对运行状况检查的响应方式来报告该环境的运行状况。Elastic Beanstalk 使用四种颜色之一来描述状态，如下表所示：

颜色	描述
灰色	正在更新您的环境。
Green	您的环境已通过最近的运行状况检查。您的环境中至少有一个实例可用并且正在接收请求。
黄色	您的环境未能通过一项或更多项运行状况检查。对您的环境的某些请求将会失败。
Red	您的环境未能通过三项或更多项的运行状况检查，或环境资源已不可用。请求持续失败。

这些描述仅适用于使用基本运行状况报告的环境。请参阅运行状况颜色和状态以了解与增强型运行状况相关的详细信息。

Elastic Load Balancing 运行状况检查

在负载均衡的环境中，Elastic Load Balancing 将每 10 秒向环境中的每个实例发送一个请求，以确认实例正常运行。默认情况下，负载均衡器配置为打开端口 80 上的 TCP 连接。如果实例确认连接，则被视为运行状况良好。

您可以选择通过在应用程序中指定现有资源来覆盖此设置。如果您指定路径 (如 `/health`)，则运行状况检查 URL 会设置为 `HTTP:80/health`。运行状况检查 URL 应设置为始终由您的应用程序提供服务的路径。如果将它设置由位于您的应用程序前面的 Web 服务器提供服务或缓存的静态页面，运行状况检查将不会显示应用程序服务器或 Web 容器的问题。有关修改运行状况检查 URL 的说明，请参阅 运行状况检查。

如果已配置运行状况检查 URL，则 Elastic Load Balancing 将预期其发送的 GET 请求返回一个 `200 OK` 响应。如果应用程序未能在 5 秒内响应或以任何其他 HTTP 状态代码响应，它将不会通过运行状况检查。连续 5 次运行状况检查失败后，Elastic Load Balancing 将禁用该实例。

有关 Elastic Load Balancing 运行状况检查的更多信息，请参阅 Elastic Load Balancing 用户指南 中的运行状况检查。

> **Note**
>
> 配置运行状况检查 URL 不会更改环境的 Auto Scaling 组的运行状况检查行为。运行状况不佳的实例将从负载均衡器中移除，但不会自动由 Amazon EC2 Auto Scaling 替换，除非您将 Amazon EC2 Auto Scaling 配置为使用 Elastic Load Balancing 运行状况检查作为替换实例的基础。要配置 Amazon EC2 Auto Scaling 以替换未通过 Elastic Load Balancing 运行状况检查的实例，请参阅Auto Scaling 运行状况检查设置。

单实例和工作线程层环境运行状况检查

在单实例或工作线程层环境中，Elastic Beanstalk 通过监控 Amazon EC2 实例状态来确定实例的运行状况。Elastic Load Balancing 运行状况设置（包括 HTTP 运行状况检查 URL）不能在这些环境类型中使用。

有关 Amazon EC2 实例状态检查的更多信息，请参阅 A mazon EC2 用户指南中的通过状态检查监控实例。

额外检查

除 Elastic Load Balancing 运行状况检查之外，Elastic Beanstalk 还监控您环境中的资源并在它们部署失败、未正确配置或变得不可用时将运行状况状态更改为红色。这些检查确认：

- 环境的 Auto Scaling 组可用且至少有一个实例。
- 环境的安全组可用且配置为允许端口 80 上的传入流量。
- 环境别名记录存在，且指向正确的负载均衡器。

- 在工作线程环境中，至少每三分钟轮询一次 Amazon Simple Queue Service (Amazon SQS) 队列。

亚马逊 CloudWatch 指标

对于基本运行状况报告，Elastic Beanstalk 服务不会向亚马逊发布任何指标。 CloudWatch用于在环境控制台的 “监控” 页面上生成图表的 CloudWatch 指标由您的环境中的资源发布。

例如，EC2 将为您环境的 Auto Scaling 组中的实例发布以下指标：

`CPUUtilization`

目前正在使用的计算单位的百分比。

`DiskReadBytes`, `DiskReadOps`, `DiskWriteBytes`, `DiskWriteOps`

读取和写入的字节数，以及读取和写入操作数。

`NetworkIn`, `NetworkOut`

发送和接收的字节数。

Elastic Load Balancing 将为您环境的负载均衡器发布以下指标：

`BackendConnectionErrors`

负载均衡器和环境实例之间失败的连接数。

`HTTPCode_Backend_2XX`, `HTTPCode_Backend_4XX`

您的环境中的实例生成的成功 (2XX) 和客户端错误 (4XX) 响应代码数。

`Latency`

负载均衡器将请求中继到实例和响应被接收之间的秒数。

`RequestCount`

完成的请求数。

这些列表并不全面。有关这些资源可以报告的指标的完整列表，请参阅《Amazon CloudWatch 开发者指南》中的以下主题：

指标

命名空间	主题
AWS::ElasticLoadBalancing::LoadBalancer	Elastic Load Balancing 指标和资源
AWS::AutoScaling::AutoScaling组	Amazon Elastic Compute Cloud 指标和资源
AWS::SQS::Queue	Amazon SQS 指标和资源
Amazon::RDS::DBInstance	Amazon RDS 维度和指标

工作线程环境运行状况指标

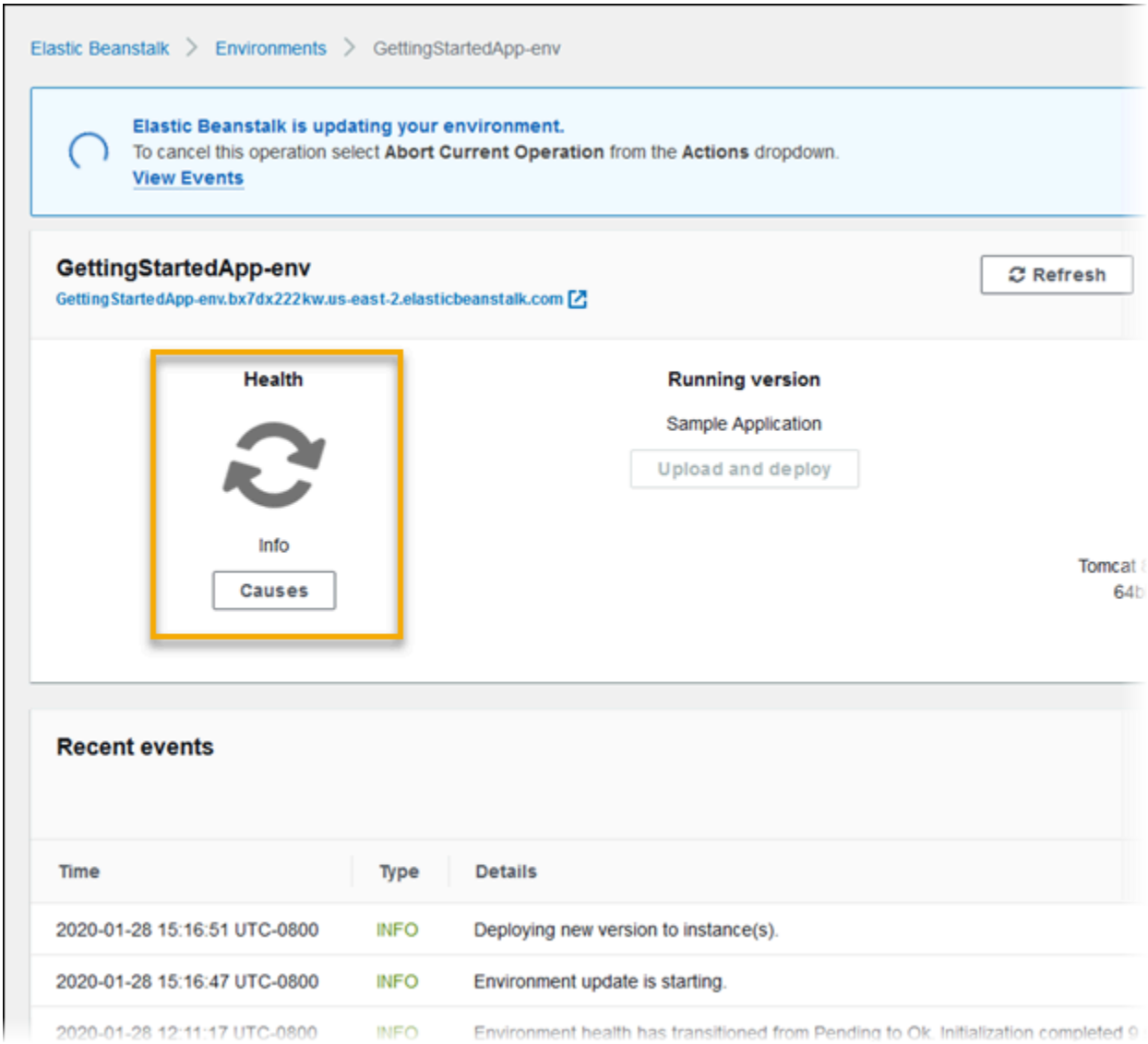
仅适用于工作线程环境，SQS 守护程序会将环境运行状况的自定义指标发布到其中 CloudWatch，值为 1 表示绿色。您可以使用`ElasticBeanstalk/SQSD`命名空间查看账户中的 CloudWatch 健康指标数据。指标维度为 `EnvironmentName`，并且指标名称为 `Health`。所有实例都会将其指标发布到同一命名空间。

要启用守护程序来发布指标，环境的实例配置文件必须有权调用 `cloudwatch:PutMetricData`。此权限包括在默认实例配置文件中。有关更多信息，请参阅管理 Elastic Beanstalk 实例配置文件。

增强型运行状况报告和监控

增强型运行状况报告是一种功能，您可以在您的环境中启用该功能以允许 AWS Elastic Beanstalk 收集有关环境中的资源的其他信息。Elastic Beanstalk 分析收集的信息，以更好地了解总体环境运行状况并帮助识别可能导致您的应用程序变得不可用的问题。

除运行状况颜色工作方式方面的更改之外，增强型运行状况还添加了状态 描述符，当环境为黄色或红色时，状态描述符提供观察到的问题的严重性指标。如果系统提供了有关当前状态的更多信息，您可以选择原因按钮，在运行状况页面上查看详细的运行状况信息。



为提供有关在您的环境中运行的 Amazon EC2 实例的运行状况详细信息，Elastic Beanstalk 将在 Amazon 系统映像 (AMI) 中为支持增强型运行状况的每个平台版本包括一个运行状况代理。运行状况代理监控 Web 服务器日志和系统指标并将它们中继到 Elastic Beanstalk 服务。Elastic Beanstalk 会分析来自 Elastic Load Balancing 和 Amazon EC2 Auto Scaling 的这些指标和数据，以提供环境运行状况的总体情况。

除收集和展示有关环境资源的信息之外，Elastic Beanstalk 还监控环境中的资源是否存在一些错误条件并向您发送通知以帮助避免失败和解决配置问题。影响环境运行状况的因素包括应用程序处理的每个请求的结果、实例操作系统的指标和最新部署的状态。

您可以使用 Elastic Beanstalk 控制台的环境概述页面或 Elastic Beanstalk 命令行界面 (EB CLI) 中的 eb health 命令实时查看运行状况。要随着时间的推移记录和跟踪环境和实例运行状况，您可以将环境配置为将 Elastic Beanstalk 收集的有关增强型运行状况报告的信息作为自定义指标发布到 Amazon

CloudWatch。针对自定义指标的 CloudWatch 费用适用于除 EnvironmentHealth（免费）之外的所有指标。

增强型运行状况报告需要版本 2 或更高的平台版本。为了监控资源和发布指标，您的环境必须同时具有实例配置文件和服务角色。如果您将 Web 服务器配置为以恰当格式提供日志，默认情况下多容器 Docker 平台将不包括 Web 服务器，但可与增强型运行状况报告结合使用。

> Windows 平台备注
>
> - 在早于版本 2 (v2) 的 Windows Server 平台版本上，此功能不可用。
> - 当您在 Windows Server 环境中启用增强型运行状况报告时，请勿更改 IIS 日志记录配置。要使增强型运行状况监控正常工作，必须使用 W3C 格式和 ETW event only (仅限 ETW 事件) 或 Both log file and ETW event (日志文件和 ETW 事件两者) 日志事件目标配置 IIS 日志记录。
>
>   此外，不要在您环境的任何实例上禁用或停止Elastic Beanstalk 运行状况代理 Windows 服务。要收集和报告实例上的增强型运行状况信息，应启用并运行该服务。

增强型运行状况要求环境具有实例配置文件。实例配置文件应包含一些角色，这些角色为您的环境实例提供收集和报告增强型运行状况信息的权限。首次在 Elastic Beanstalk 控制台中使用 v2 平台版本创建环境时，Elastic Beanstalk 提示您创建所需的角色，并默认启用增强型运行状况报告。继续阅读有关增强型运行状况报告的工作方式的详细信息，或参阅启用 Elastic Beanstalk 增强型运行状况报告立即开始使用增强型运行状况报告。

Amazon Linux 2 平台需要实例配置文件，以便无条件支持增强型运行状况。当您使用 Amazon Linux 2 平台创建环境时，Elastic Beanstalk 始终启用增强型运行状况。无论您如何创建环境（使用 Elastic Beanstalk 控制台、EB CLI、AWS CLI 或 API），都会发生这种情况。

主题

- 启用 Elastic Beanstalk 增强型运行状况报告
- 采用环境管理控制台的增强型运行状况监控
- 运行状况颜色和状态
- 实例指标
- 为环境配置增强型运行状况规则
- 发布环境的 Amazon CloudWatch 自定义指标
- 将增强型运行状况报告与 Elastic Beanstalk API 结合使用
- 增强型运行状况日志格式
- 通知和问题排查

Elastic Beanstalk 运行状况代理

Elastic Beanstalk 运行状况代理是在您的环境中的每个 Amazon EC2 实例上运行的守护程序进程（或 Windows 环境上的一项服务），它可监控操作系统和应用程序级别的运行状况指标并将问题报告给 Elastic Beanstalk。从每个平台的 2.0 版本开始，所有平台版本中都包含运行状况代理。

运行状况代理将报告与由 Amazon EC2 Auto Scaling 和 Elastic Load Balancing 作为基本运行状况报告的一部分发布到 CloudWatch 的那些指标相似的指标，包括 CPU 负载、HTTP 代码和延迟。不过，运行状况代理会以与基本运行状况报告相比更小的粒度和更高的频率直接向 Elastic Beanstalk 报告。

对于基本运行状况，这些指标每五分钟发布一次，并且可通过环境管理控制台中的图表监控。对于增强型运行状况，Elastic Beanstalk 运行状况代理每 10 秒向 Elastic Beanstalk 报告指标一次。Elastic Beanstalk 使用运行状况代理提供的指标来确定环境中每个实例的运行状况状态，并将这些指标与其他因素结合使用来确定环境的总体运行状况。

环境的总体运行状况可在 Elastic Beanstalk 控制台的环境概述页面上实时查看，并由 Elastic Beanstalk 每 60 秒向 CloudWatch 发布一次。您可以使用 EB CLI 中的 eb health 命令实时查看运行状况代理报告的详细指标。

通过支付额外费用，您可以选择每 60 秒将单个实例和环境级别的指标发布到 CloudWatch 一次。然后，可使用已发布到 CloudWatch 的指标在环境管理控制台中创建监控图表。

增强型运行状况报告仅在您选择将增强型运行状况指标发布到 CloudWatch 时产生费用。在使用增强型运行状况时，您仍可免费发布基本运行状况指标，即使您不选择发布增强型运行状况指标。

请参阅实例指标以了解有关运行状况代理报告的指标的详细信息。有关将增强型运行状况指标发布到 CloudWatch 的详细信息，请参阅发布环境的 Amazon CloudWatch 自定义指标。

实例和环境运行状况的确定因素

除基本运行状况报告系统检查（包括Elastic Load Balancing 运行状况检查和资源监控）之外，Elastic Beanstalk 增强型运行状况报告还收集有关您的环境中的实例状态的其他数据。这包括操作系统指标、服务器日志以及正在进行的环境操作（如部署和更新）的状态。Elastic Beanstalk 运行状况报告服务将来自所有可用源的信息合并在一起，并分析这些信息以确定环境的总体运行状况。

操作和命令

当您对环境执行某项操作（如部署新版本的应用程序）时，Elastic Beanstalk 将进行多项影响环境的运行状况状态的更改。

例如，如果您将某个应用程序的新版本部署到正在运行多个实例的环境中，您可能会在使用 EB CLI 监控环境的运行状况时看到如下类似的消息。

```
  id              status     cause
    Overall       Info       Command is executing on 3 out of 5 instances
  i-bb65c145      Pending    91 % of CPU is in use. 24 % in I/O wait
                             Performing application deployment (running for 31 seconds)
  i-ba65c144      Pending    Performing initialization (running for 12 seconds)
  i-f6a2d525      Ok         Application deployment completed 23 seconds ago and took 26
seconds
  i-e8a2d53b      Pending    94 % of CPU is in use. 52 % in I/O wait
                             Performing application deployment (running for 33 seconds)
  i-e81cca40      Ok
```

在此示例中，环境的总体状态为 Ok，此状态的原因是 Command is executing on 3 out of 5 instances。环境中的三个实例的状态为等待，表示一项操作正在进行。

当一项操作完成时，Elastic Beanstalk 将报告有关该操作的额外信息。对于本示例，Elastic Beanstalk 显示了有关一个已使用应用程序的新版本进行更新的实例的以下信息：

```
i-f6a2d525      Ok         Application deployment completed 23 seconds ago and took 26
 seconds
```

实例运行状况信息还包括有关环境中每个实例的最新部署的详细信息。每个实例都会报告一个部署 ID 和状态。部署 ID 是一个整数，每次在您部署新版本的应用程序或针对环境变量等有关实例的配置选项更改设置时，其数量就会增加一个。滚动部署失败后，您可以使用部署信息来识别那些运行错误应用程序版本的实例。

在原因列中，Elastic Beanstalk 包括有关成功的操作以及在多个运行状况检查中的其他运行状况状态的信息性消息，但它们不会无限期保持。运行状况不佳的环境状态的原因将持续，直到环境恢复运行状况良好状态。

命令超时

Elastic Beanstalk 从操作开始的时间起应用一个命令超时，以允许实例转换为运行状况良好状态。此命令超时在您的环境的更新和部署配置中设置（在 aws:elasticbeanstalk:command 命名空间中），默认为 10 分钟。

在滚动更新期间，Elastic Beanstalk 将为操作中每个批次应用不同的超时。此超时将作为环境的滚动更新配置的一部分设置（在 aws:autoscaling:updatepolicy:rollingupdate 命名空间中）。如果在滚动更新超时期间，该批次中所有实例都运行状况良好，则操作将继续到下一个批次。否则，操作将失败。

> **Note**
>
> 如果应用程序未通过运行状况检查（即未处于 OK (正常) 状态），但在另一级别很稳定，您可以在 `HealthCheckSuccessThreshold`中设置 `aws:elasticbeanstalk:command namespace` 选项，从而更改 Elastic Beanstalk 将实例视为运行状况良好的级别。

一个 Web 服务器环境若要被视为运行状况良好，该环境或批次中每个实例必须在两分钟内连续通过 12 次运行状况检查。对于工作线程层环境，每个实例必须通过 18 次运行状况检查。在命令超时之前，Elastic Beanstalk 不会在运行状况检查失败时降低环境的运行状况状态。如果环境中的实例在命令超时范围内转变为运行状况良好，则操作成功。

HTTP 请求

当没有对环境正在进行的操作时，有关实例和环境运行状况的主要信息源为每个实例的 Web 服务器日志。为了确定实例的运行状况和环境的总体运行状况，Elastic Beanstalk 将考虑请求的数量、每个请求的结果以及解决每个请求的速度。

在基于 Linux 的平台上，Elastic Beanstalk 将读取并解析 Web 服务器日志以获取有关 HTTP 请求的信息。在 Windows Server 平台上，Elastic Beanstalk 将直接从 IIS Web 服务器接收此信息。

您的环境可能不具有有效的 Web 服务器。例如，多容器 Docker 平台不包含 Web 服务器。其他平台包括一个 Web 服务器，并且您的应用程序可能会将其禁用。在这些情况下，您的环境需要额外的配置，以向 Elastic Beanstalk 运行状况代理提供日志，日志格式为将运行状况信息中继到 Elastic Beanstalk 服务所需的格式。有关更多信息，请参阅 增强型运行状况日志格式。

操作系统指标

Elastic Beanstalk 监控运行状况代理报告的操作系统指标以识别系统资源始终较少的实例。

请参阅实例指标以了解有关运行状况代理报告的指标的详细信息。

运行状况检查规则自定义

Elastic Beanstalk 增强型运行状况报告依靠一组规则来确定环境的运行状况。其中一些规则可能不适合您的特定应用程序。一种常见情况是应用程序设计为频繁返回 HTTP 4xx 错误。Elastic Beanstalk 使用其默认规则之一判断出现问题，并根据错误比率将您环境的运行状况从“正常”更改为“警告”、“已降级”或“严重”。为正确处理这种情况，Elastic Beanstalk 允许您配置此规则并忽略应用程序 HTTP 4xx 错误。有关详细信息，请参阅为环境配置增强型运行状况规则。

增强型运行状况角色

增强型运行状况报告需要两个角色：一个适用于 Elastic Beanstalk 的服务角色和一个适用于环境的实例配置文件。服务角色允许 Elastic Beanstalk 代表您与其他 AWS 服务进行交互，以收集有关环境中的资源的信息。实例配置文件允许环境中的实例向 Amazon S3 写入日志并向 Elastic Beanstalk 服务传递增强型运行状况信息。

使用 Elastic Beanstalk 控制台或 EB CLI 创建 Elastic Beanstalk 环境时，Elastic Beanstalk 会创建默认服务角色并将所需的托管策略附加到您环境的默认实例配置文件。

如果您使用 API、开发工具包或 AWS CLI 创建环境，则必须提前创建这些角色，并在创建环境时指定它们，然后才能使用增强型运行状况。有关为您的环境创建适当角色的说明，请参阅服务角色、实例配置文件和用户策略。

我们建议您对实例配置文件和服务角色使用托管策略。托管策略是 Elastic Beanstalk 维护的 AWS Identity and Access Management (IAM) 策略。使用托管策略可确保您的环境具有其正常运行所需的所有权限。

对于实例配置文件，您可以分别对 Web 服务器层或工作线程层环境使用 `AWSElasticBeanstalkWebTier` 或 `AWSElasticBeanstalkWorkerTier` 托管策略。有关这两个托管实例配置文件策略的详细信息，请参阅 the section called “实例配置文件”。

增强型运行状况授权

Elastic Beanstalk 实例配置文件托管策略包含 `elasticbeanstalk:PutInstanceStatistics` 操作的权限。此操作不是 Elastic Beanstalk API 的一部分。这是不同的 API 的一部分，环境实例在内部使用它向 Elastic Beanstalk 服务传递增强型运行状况信息。您不直接调用此 API。

创建新环境时，默认状态下已启用 `elasticbeanstalk:PutInstanceStatistics` 操作的授权。为了提高环境的安全性并帮助防止以您的名义骗取运行状况数据，我们建议您启用该操作的授权。如果您对实例配置文件使用托管策略，则此功能将可在您的新环境中使用，而无需进行进一步配置。如果您使用自定义实例配置文件而不是托管策略，则您的环境可能会显示 No Data（无数据）运行状况状态。发生这种情况的原因是，实例未获授权执行将增强型运行状况数据传送到服务的操作。

要授权执行此操作，请在您的实例配置文件中包含以下语句。

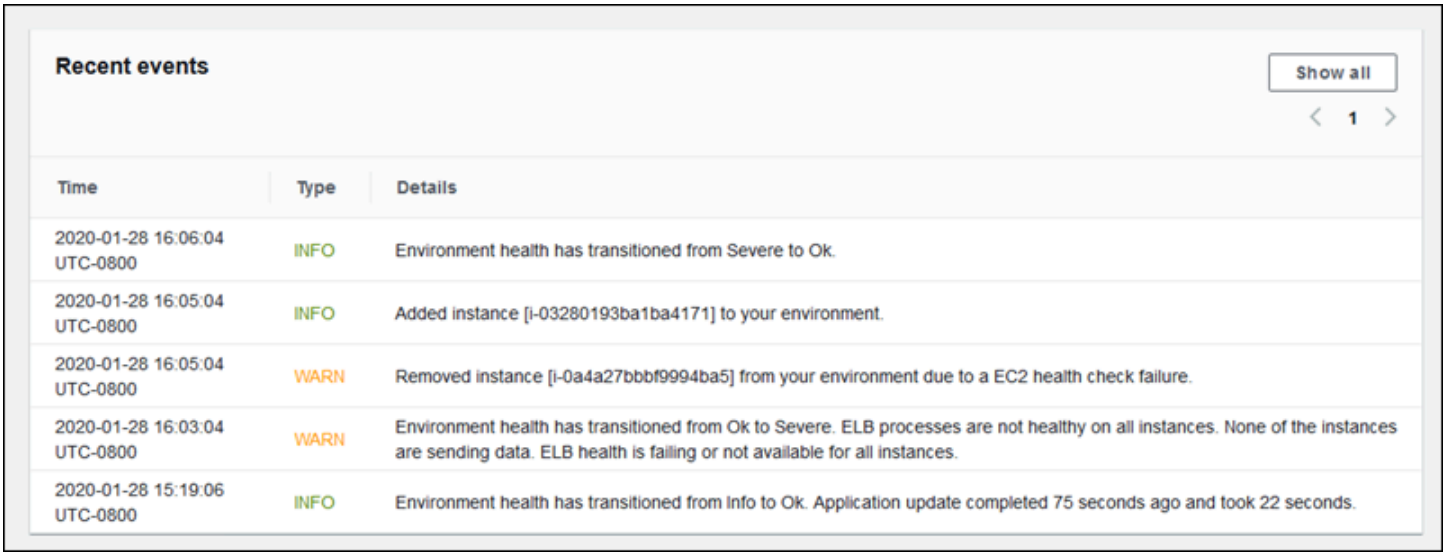
```
{
  "Sid": "ElasticBeanstalkHealthAccess",
  "Action": [
    "elasticbeanstalk:PutInstanceStatistics"
  ],
  "Effect": "Allow",
  "Resource": [
    "arn:aws:elasticbeanstalk:*:*:application/*",
    "arn:aws:elasticbeanstalk:*:*:environment/*"
  ]
}
```

如果您目前不想使用增强型运行状况授权，则可以将 the section called “aws:elasticbeanstalk:healthreporting:system” 命名空间中的 `EnhancedHealthAuthEnabled` 选项设为 `false`。如果此选项处于禁用状态，则不需要前面所述的权限。您可以从实例配置文件中删除它们，以获得对应用程序和环境的最小权限访问。

> **Note**
>
> 以前 `EnhancedHealthAuthEnabled` 默认设置为 `false`，这会导致默认情况下，`elasticbeanstalk:PutInstanceStatistics` 操作的授权也会遭禁用。要为现有环境启用此操作，请将 the section called “aws:elasticbeanstalk:healthreporting:system” 命名空间中的 `EnhancedHealthAuthEnabled` 选项设为 `true`。您可以通过在配置文件中使用选项设置来配置此选项。

增强型运行状况事件

当环境在不同状态间转换时，增强型运行状况系统将生成事件。以下示例显示了环境在信息、正常和严重状态间转换时输出的事件。



转换到较差的状态时，增强型运行状况事件将包括一条消息，指示转换原因。

并非所有实例级别的状态更改都会导致 Elastic Beanstalk 发送事件。为了防止错误警报，仅当一个问题在多次检查中连续出现时，Elastic Beanstalk 才生成与运行状况相关的事件。

实时的环境级别运行状况信息（包括状态、颜色和原因）可在 Elastic Beanstalk 控制台和 EB CLI 的环境概述页面中找到。通过将 EB CLI 连接到您的环境并运行 eb health 命令，您还可以查看来自环境中每个实例的实时状态。

更新、部署和扩展期间的增强型运行状况报告行为

启用增强型运行状况报告可影响您的环境在配置更新和部署期间的行为。在所有实例一致通过运行状况检查之前，Elastic Beanstalk 不会完成批量更新。此外，由于增强型运行状况报告将对运行状况应用更高标准并监控更多因素，因此通过了基本运行状况报告的 ELB 运行状况检查的实例不一定通过增强型运行状况报告的检查。请参阅有关滚动配置更新和滚动部署的主题，以了解有关运行状况检查如何影响更新过程的详细信息。

增强型运行状况报告还可以凸显为 Elastic Load Balancing 设置恰当的运行状况检查 URL 的必要性。当您的环境纵向扩展以满足需求时，新的实例在通过足够的 ELB 运行状况检查之后便会立即开始接收请求。如果没有配置运行状况检查 URL，则这可能是在新实例能够接受 TCP 连接之后的短短 20 秒。

如果您的应用程序在负载均衡器声明其运行状况足够良好并可以接受流量之前还没有完成启动，您将会看到一大批失败的请求，而且您的环境将开始让运行状况检查失败。命中您的应用程序所服务的路径的运行状况检查 URL 可防止此问题。在对运行状况检查 URL 的 GET 请求返回 200 状态代码之前，ELB 运行状况检查不会通过。

启用 Elastic Beanstalk 增强型运行状况报告

使用最新平台版本创建的新环境包含支持增强型运行状况报告的 AWS Elastic Beanstalk 运行状况代理。如果您在 Elastic Beanstalk 控制台中或使用 EB CLI 创建环境，则默认启用增强型运行状况。您也可以使用配置文件在应用程序的源代码中设置运行状况报告首选项。

增强型运行状况报告要求实例配置文件和有标准权限集的服务角色。在 Elastic Beanstalk 控制台中创建环境时，Elastic Beanstalk 会自动创建所需的角色。有关创建您的第一个环境的说明，请参阅开始使用 Elastic Beanstalk。

主题

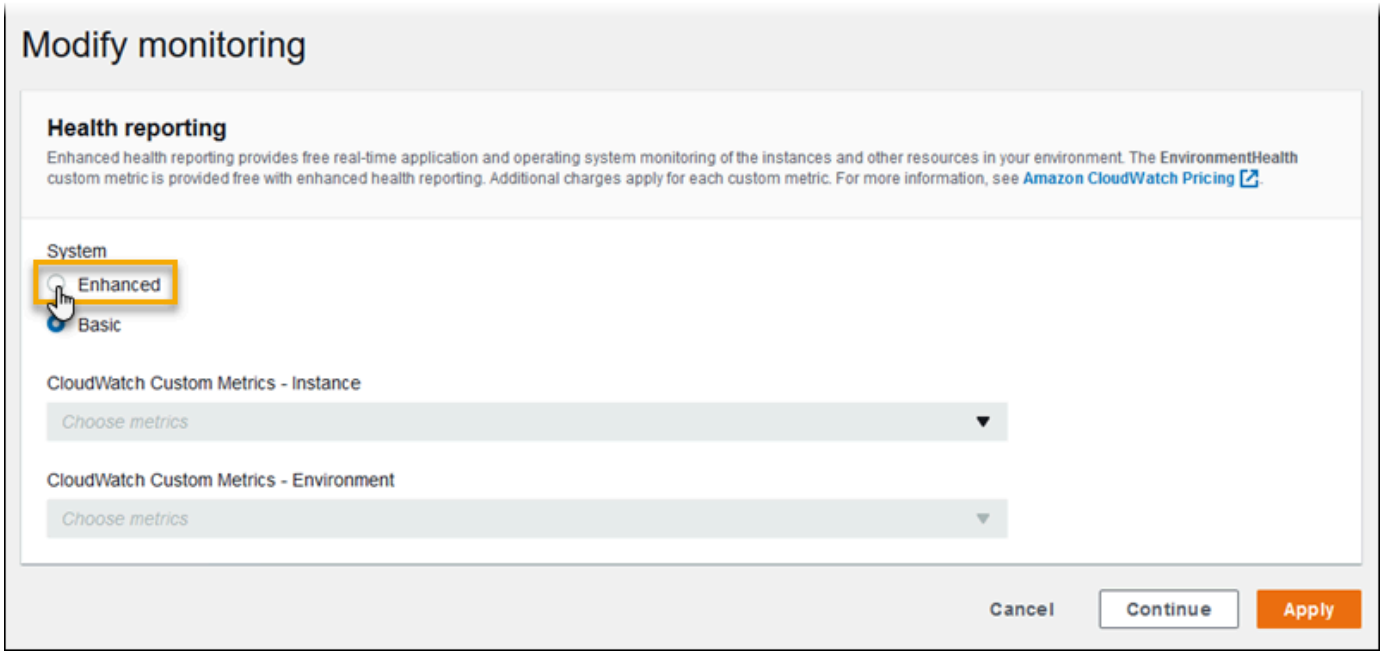
使用 Elastic Beanstalk 控制台启用增强型运行状况报告

使用 Elastic Beanstalk 控制台在正在运行的环境中启用增强型运行状况报告

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Monitoring (监控) 配置类别中，选择 Edit (编辑)。
5. 在运行状况报告下，为系统选择增强型。



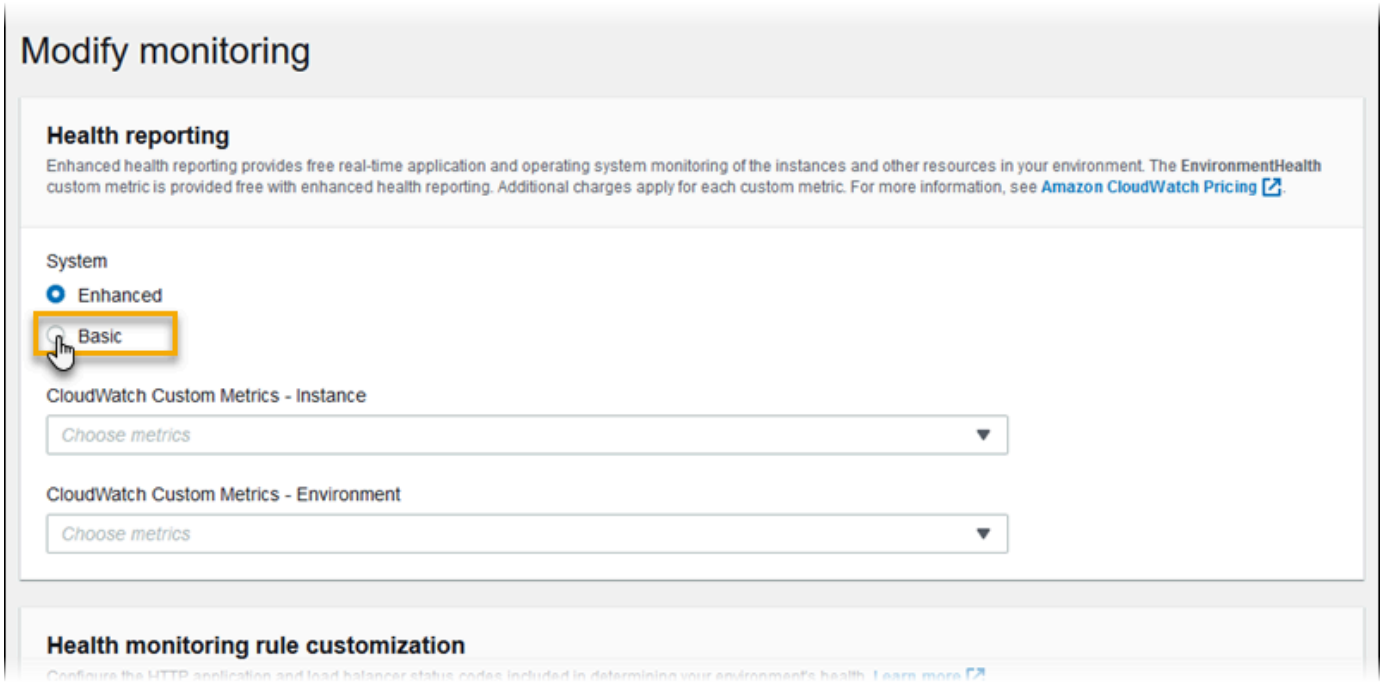
> **Note**
>
> 如果您使用的是不受支持的平台或版本，则不会显示用于增强型运行状况报告的选项。

6. 要保存更改，请选择页面底部的 Apply（应用）。

在使用版本 2 (v2) 平台版本创建新环境时，Elastic Beanstalk 控制台默认使用增强型运行状况报告。您可以通过在环境创建期间更改运行状况报告选项来禁用增强型运行状况报告。

在使用 Elastic Beanstalk 控制台创建环境时禁用增强型运行状况报告

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 创建一个应用程序或选择现有应用程序。
3. 创建一个环境。在创建新环境页面上，在选择创建环境之前，选择配置更多选项。
4. 在 Monitoring (监控) 配置类别中，选择 Edit (编辑)。
5. 在运行状况报告下，为系统选择基本。



6. 选择 Save（保存）。

使用 EB CLI 启用增强型运行状况报告

在使用 eb create 命令创建新环境时，EB CLI 默认启用增强型运行状况报告并应用默认实例配置文件和服务角色。

您可以使用 `--service-role` 选项通过名称指定不同的服务角色。

如果您有一个在 v2 平台版本上使用基本运行状况报告运行的环境且希望将其切换到增强型运行状况，请按以下步骤操作。

使用 EB CLI 在正在运行的环境上启用增强型运行状况

1. 使用 eb config 命令在默认文本编辑器中打开配置文件。

```
~/project$ eb config
```

2. 在设置部分找到 `aws:elasticbeanstalk:environment` 命名空间。确保 `ServiceRole` 的值不为空且与您的服务角色的名称匹配。

```
  aws:elasticbeanstalk:environment:
    EnvironmentType: LoadBalanced
    ServiceRole: aws-elasticbeanstalk-service-role
```

3. 在 aws:elasticbeanstalk:healthreporting:system: 命名空间下，将 SystemType 的值更改为 **enhanced**。

```
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
```

4. 保存配置文件，关闭文本编辑器。
5. EB CLI 将启动环境更新以应用配置更改。等待操作完成或按 Ctrl+C 以安全退出。

```
~/project$ eb config
Printing Status:
INFO: Environment update is starting.
INFO: Health reporting type changed to ENHANCED.
INFO: Updating environment no-role-test's configuration settings.
```

使用配置文件启用增强型运行状况报告

您可以通过在源包中包含配置文件来启用增强型运行状况报告。下面示例的配置文件启用增强型运行状况报告并为环境分配默认服务和实例配置文件：

Example .ebextensions/enhanced-health.config

```
option_settings:
  aws:elasticbeanstalk:healthreporting:system:
    SystemType: enhanced
  aws:autoscaling:launchconfiguration:
    IamInstanceProfile: aws-elasticbeanstalk-ec2-role
  aws:elasticbeanstalk:environment:
    ServiceRole: aws-elasticbeanstalk-service-role
```

如果您创建了自己的实例配置文件或服务角色，请将突出显示的文本替换为这些角色的名称。

采用环境管理控制台的增强型运行状况监控

在 AWS Elastic Beanstalk 中启用增强型运行状况报告后，您可以在环境管理控制台中监控环境运行状况。

主题

- 环境概述

- 环境运行状况页面
- 监控页面

环境概述

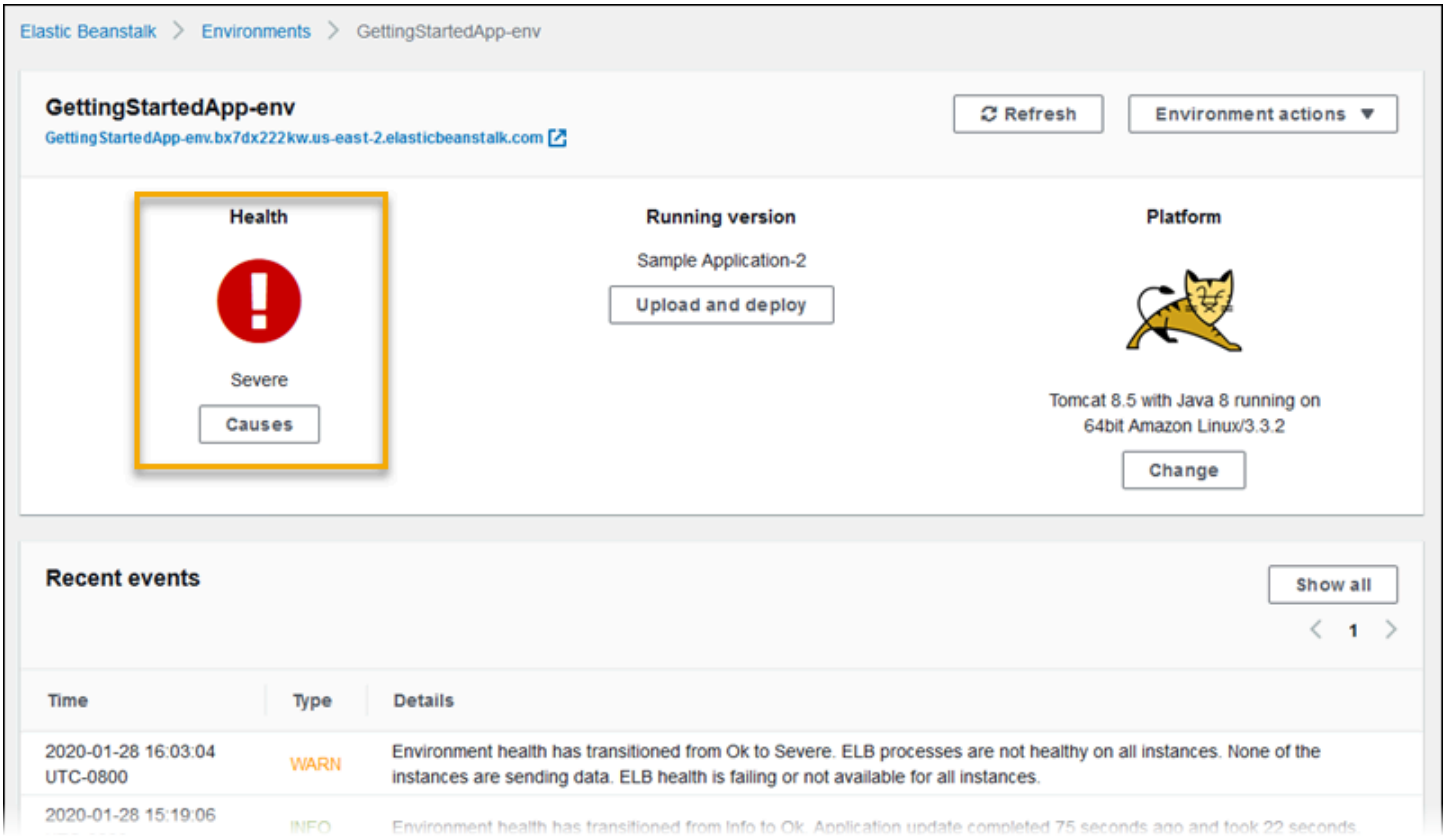
环境概述会显示环境的运行状况，并列出可提供有关运行状况的最新更改信息的事件。

查看环境概述

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

有关当前环境运行状况的详细信息，请选择 Causes (原因) 来打开 Health (运行状况) 页面。或者，在导航窗格中，选择 Health (运行状况)。



环境运行状况页面

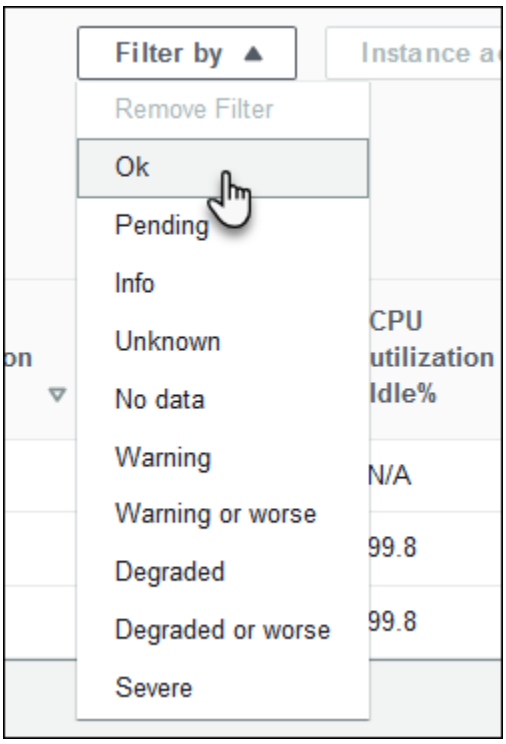
Health (运行状况) 页面会显示环境以及其中每个 Amazon EC2 实例的运行状况、指标和故障原因。

> **Note**
>
> 只有当您为环境启用了增强型运行状况监控时，Elastic Beanstalk 才会显示 Health（运行状况）页面。

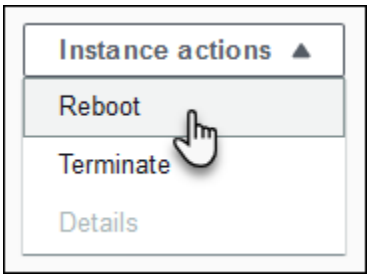
下图显示了 Linux 环境的 Health (运行状况) 页面。

下图显示了 Windows 环境 Health (运行状况) 页面。请注意，CPU 指标与 Linux 环境中的相应指标不同。

在页面顶部，您可以看到环境实例的总数以及每种状态所对应的实例数。要仅显示处于特定状态的实例，请选择 Filter By (筛选依据)，然后选择一种状态。



要重启或终止运行不正常的实例，请选择 Instance Actions (实例操作)，然后选择 Reboot (重启) 或 Terminate (终止)。



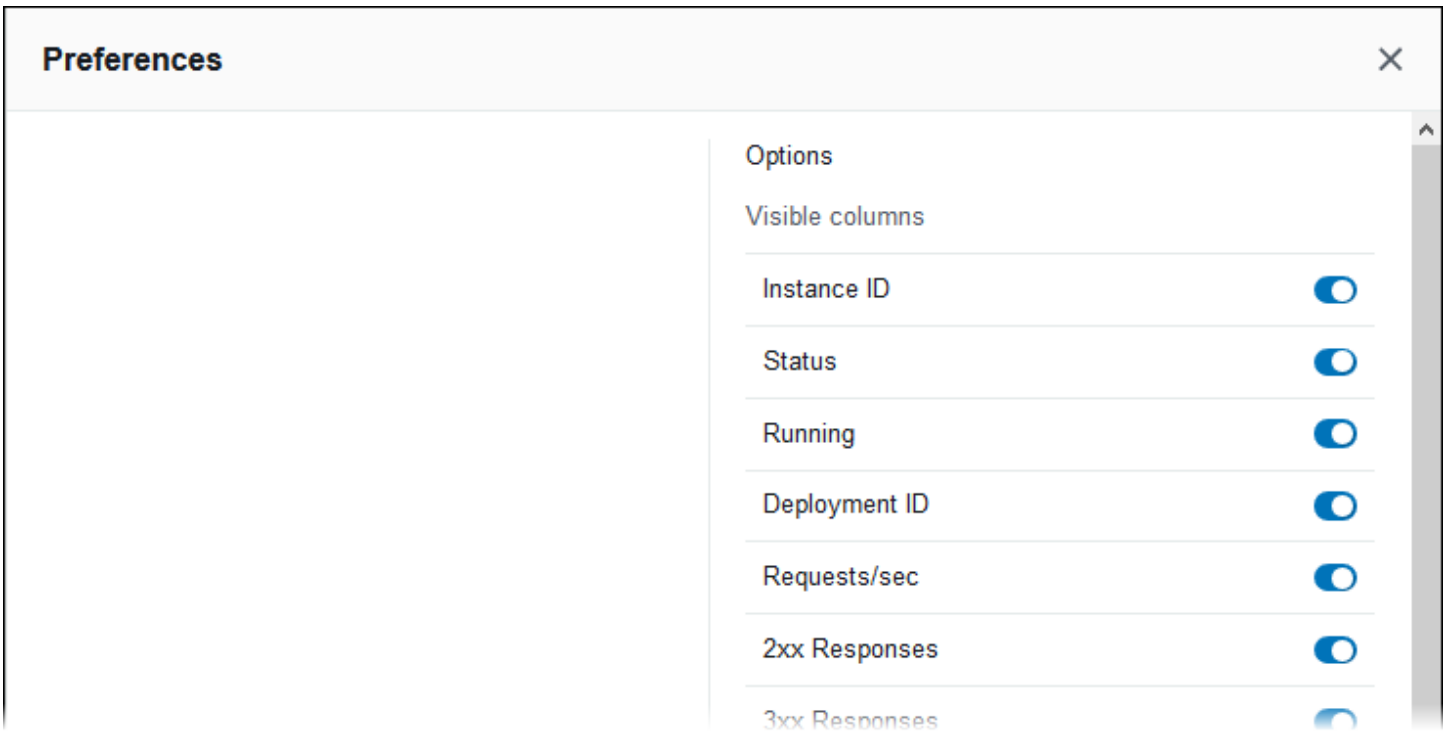
Elastic Beanstalk 每 10 秒更新一次 Health (运行状况) 页面。它报告了有关环境和实例运行状况的信息。

对于环境中的每个 Amazon EC2 实例，此页面会显示实例的 ID 和状态、实例启动后的时间、在实例上执行的最新部署的 ID、实例服务于的请求的响应和延迟，以及负载和 CPU 利用率信息。Overall (总体) 行显示整个环境的平均响应和延迟信息。

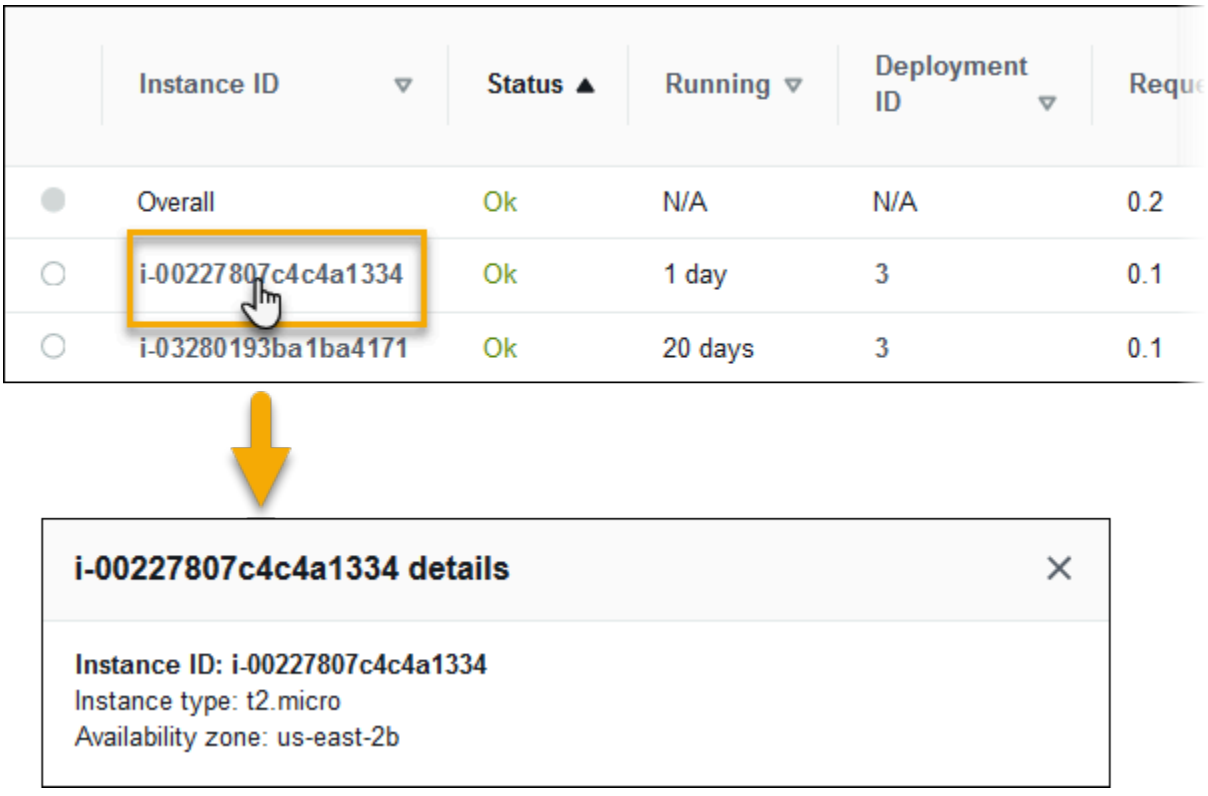
此页面在一个非常宽的表中显示了许多详细信息。要隐藏某些列，请选择

(Prefe

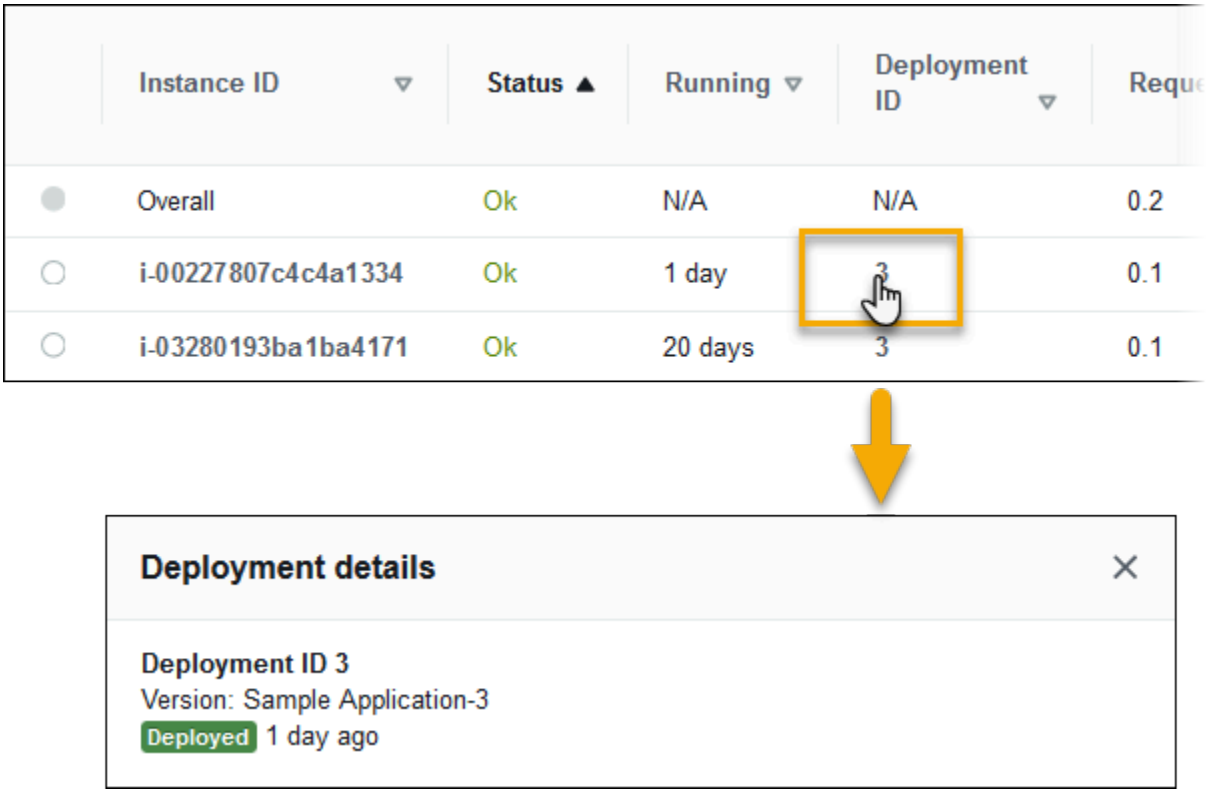
(首选项))。选择或清除列名称，然后选择 Confirm (确认)。



选择任意实例的 Instance ID (实例 ID) 可查看有关该实例的更多信息，包括其可用区和实例类型。



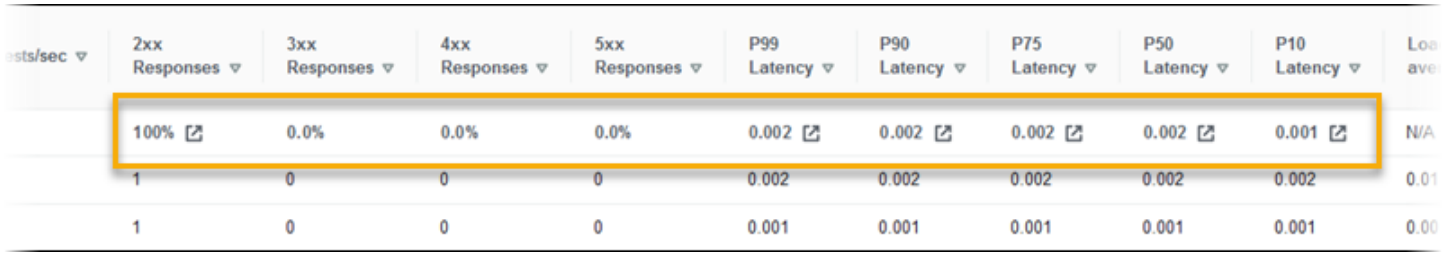
选择任意实例的 Deployment ID (部署 ID) 可查看有关针对实例的上次部署的信息。



部署信息包括以下内容：

- 部署 ID - 部署的唯一标识符。部署 ID 从 1 开始，每次在您部署新的应用程序版本或者更改会影响环境中的实例上运行的软件或操作系统的配置设置时，部署 ID 就会逐一往上增加。
- 版本 - 部署中所用应用程序源代码的版本标签。
- 状态 - 部署状态，可以是 `In Progress`、`Deployed` 或 `Failed`。
- 时间 - 对于正在进行的部署，表示部署开始时间。对于已完成的部署，表示部署结束时间。

如果您在环境中启用了 X-Ray 集成，并使用 AWS X-Ray 开发工具包来检测应用程序，则 Health (运行状况) 页面会向 AWS X-Ray 控制台的概览行中添加链接。



选择链接可查看与 AWS X-Ray 控制台中突出显示的统计数据有关的跟踪信息。

监控页面

Monitoring (监控) 页面会显示增强型运行状况报告系统生成的自定义 Amazon CloudWatch 指标的汇总统计数据和图表。有关将图表和统计数据添加到此页面的说明，请参阅在AWS管理控制台中监控环境运行状况。

运行状况颜色和状态

与基本运行状况报告类似，增强型运行状况报告使用四种颜色来代表实例和总体环境运行状况。增强型运行状况报告还提供七种运行状况状态，这些单个词的描述符将更好地指明您的环境的状态。

实例状态和环境状态

每次 Elastic Beanstalk 对您的环境运行运行状况检查时，增强型运行状况报告将通过分析所有可用数据来检查您的环境中每个实例的运行状况。如果任何较低级别的检查未通过，则 Elastic Beanstalk 将降低实例的运行状况级别。

Elastic Beanstalk 在环境管理控制台中显示有关总体环境的运行状况信息（颜色、状态和原因）。此信息也可在 EB CLI 中找到。各个实例的运行状况状态和原因消息每 10 秒更新一次，并且，当使用 eb health 查看运行状况时，这些信息会在 EB CLI 中提供。

Elastic Beanstalk 使用实例运行状况中的更改来评估环境运行状况，但不会立即更改环境运行状况状态。如果一个实例在任意的一分钟时间段内至少三次未通过运行状况检查，则 Elastic Beanstalk 会将环境的运行状况降级。根据环境中实例的数量和识别的问题，一个运行状况不佳的实例可能会导致 Elastic Beanstalk 显示一条信息性消息，或导致环境的运行状况状态从绿色（OK (正常)）更改为黄色（Warning (警告)）或红色（Degraded (降级) 或 Severe (严重)）。

正常 (绿色)

在以下情况下显示此状态：

- 一个实例通过了运行状况检查，且运行状况代理没有报告任何问题。
- 环境中的大多数实例通过了运行状况检查，且运行状况代理没有报告重大问题。
- 一个实例通过了运行状况检查并正常完成请求。

示例：您的环境最近进行了部署，并且正在正常接收请求。5% 的请求返回 400 系列错误。对每个实例的部署正常完成。

消息（实例）：应用程序部署已于 23 秒前完成，耗时 26 秒。

警告 (黄色)

在以下情况下显示此状态：

- 运行状况代理报告了实例或环境的中等数量的请求失败或其他问题。
- 正在对一个实例进行一项操作且耗费很长时间。

示例：环境中的一个实例的状态为严重。

消息（环境）：5 个实例中有 1 个实例的服务受损。

降级 (红色)

当运行状况代理报告了实例或环境的大量的请求失败或其他问题时，显示此状态。

示例：环境处于扩展到 5 个实例的过程中。

消息（环境）：4 个活动实例低于 Auto Scaling 组的最小大小 5。

严重 (红色)

当运行状况代理报告了实例或环境的非常大量的请求失败或其他问题时，显示此状态。

示例：Elastic Beanstalk 无法联系负载均衡器来获取实例运行状况。

消息（环境）：ELB 运行状况失败或对所有实例均不可用。任何实例都未发送数据。无法代入角色“arn:aws:iam::123456789012:role/aws-elasticbeanstalk-service-role”。确认角色存在并已正确配置。

消息（实例）：实例 ELB 运行状况已在 37 分钟的时间内不可用。没有数据。上次显示还是在 37 分钟前。

信息 (绿色)

在以下情况下显示此状态：

- 正在对一个实例进行一项操作。
- 正在对环境中几个实例进行一项操作。

示例：正在将新的应用程序版本部署到正在运行的实例。

消息（环境）：正在对 5 个实例中的 3 个实例执行命令。

消息（实例）：正在执行应用程序部署（运行了 3 秒）。

等待 (灰色)

在命令超时范围内正在对一个实例进行一项操作时，显示此状态。

示例：您最近创建了环境，实例正在进行引导启动。

消息：正在执行初始化（运行了 12 秒）。

未知 (灰色)

Elastic Beanstalk 和运行状况代理报告实例上的数据量不足时显示此状态。

示例：未接收数据。

已暂停（灰色）

当 Elastic Beanstalk 停止监控环境的运行状况时，显示此状态。环境可能无法正常工作。某些严重运行状况（如果持续较长时间）会导致 Elastic Beanstalk 将环境转换为暂停状态。

示例：Elastic Beanstalk 无法访问环境的服务角色。

示例：Elastic Beanstalk 为环境创建的 Auto Scaling 组已被删除。

消息：环境运行状况已从正常转换为严重。没有实例。Auto Scaling 组所需的容量已设置为 1。

实例指标

实例指标提供有关环境中实例运行状况的信息。每个实例上都有 Elastic Beanstalk 运行状况代理运行。此代理收集有关实例的指标并将指标中继到 Elastic Beanstalk，后者分析这些指标以确定您环境中实例的运行状况。

实例上 Elastic Beanstalk 运行状况代理从 Web 服务器和操作系统中收集有关实例的指标。为获取有关基于 Linux 的平台的 Web 服务器信息，Elastic Beanstalk 将读取并解析 Web 服务器日志。在 Windows Server 平台上，Elastic Beanstalk 将直接从 IIS Web 服务器接收此信息。Web 服务器提供有关传入 HTTP 请求的信息：传入请求的数量、出错请求的数量以及解决错误所花时间。操作系统提供有关实例资源状态的快照信息：CPU 负载及每种处理类型的用时分布。

运行状况代理收集 Web 服务器和操作系统指标，并且每 10 秒将它们中继到 Elastic Beanstalk。Elastic Beanstalk 分析这些数据并使用分析结果来更新每个实例和环境的运行状况状态。

主题

- Web 服务器指标
- 操作系统指标
- Windows Server 上的 IIS 中的 Web 服务器指标捕获

Web 服务器指标

在基于 Linux 的平台上，Elastic Beanstalk 运行状况代理从日志（由处理针对环境中每个实例的请求的 Web 容器或服务器生成）读取 Web 服务器指标。Elastic Beanstalk 平台配置为生成两个日志：一个为人类可读的格式，另一个为机器可读的格式。运行状况代理每 10 秒将机器可读的日志中继到 Elastic Beanstalk 一次。

有关 Elastic Beanstalk 使用的日志格式的更多信息，请参阅增强型运行状况日志格式。

在 Windows Server 平台上，Elastic Beanstalk 将一个模块添加到 IIS Web 服务器的请求管道并捕获有关 HTTP 请求时间和响应代码的指标。此模块使用高性能进程间通信 (IPC) 通道将这些指标发送到实例上的运行状况代理。有关实施详细信息，请参阅Windows Server 上的 IIS 中的 Web 服务器指标捕获。

报告的 Web 服务器指标

`RequestCount`

过去 10 秒内 Web 服务器每秒处理的请求数。在 EB CLI 和环境运行状况页面中显示为平均 `r/sec`（请求数/秒）。

`Status2xx`, `Status3xx`, `Status4xx`, `Status5xx`

过去 10 秒内导致每种类型的状态代码的请求数。例如，成功请求返回 200 OK；重定向为 301；如果输入的 URL 与应用程序中任何资源不匹配，则返回 404。

EB CLI 和环境运行状况页面以实例请求数和占环境总体请求数的百分比的形式显示这些指标。

`p99.9`, `p99`, `p95`, `p90`, `p85`, `p75`, `p50`, `p10`

过去 10 秒内最慢的 x% 的请求的平均延迟，其中 x 是数字与 100 之差。例如，`p99 1.403` 表示过去 10 秒内最慢的 1% 请求的平均延迟为 1.403 秒。

操作系统指标

Elastic Beanstalk 运行状况代理将报告以下操作系统指标。Elastic Beanstalk 使用这些指标确定持续承受重负载的实例。这些指标因操作系统而异。

报告的操作系统指标 - Linux

`Running`

自启动实例以来经过的时间量。

`Load 1`, `Load 5`

过去一分钟和五分钟时段内的负载平均值。显示为一位小数值，表示该时间段内正在运行的进程的平均数。如果显示的数字大于可用的 vCPU (线程) 数，则另一个数为正在等待的进程的平均数。

例如，如果您的实例类型有四个 vCPU，且负载为 4.5，则该时间段内平均有 0.5 个进程在等待，相当于有一个进程等待 50% 的时间。

`User %`, `Nice %`, `System %`, `Idle %`, `I/O Wait %`

过去 10 秒内 CPU 在每个状态中耗费的时间的百分比。

报告的操作系统指标 - Windows

`Running`

自启动实例以来经过的时间量。

`% User Time`, `% Privileged Time`, `% Idle Time`

过去 10 秒内 CPU 在每个状态中耗费的时间的百分比。

Windows Server 上的 IIS 中的 Web 服务器指标捕获

在 Windows Server 平台上，Elastic Beanstalk 将一个模块添加到 IIS Web 服务器的请求管道并捕获有关 HTTP 请求时间和响应代码的指标。此模块使用高性能进程间通信 (IPC) 通道将这些指标发送到实例上的运行状况代理。运行状况代理聚合这些指标，将它们与操作系统指标组合，然后将组合后的指标发送到 Elastic Beanstalk 服务。

实施详情

为从 IIS 捕获指标，Elastic Beanstalk 实施托管的 `IHttpModule` 并订阅 `BeginRequest` 和 `EndRequest` 事件。这使此模块报告由 IIS 处理的所有 Web 请求的 HTTP 请求延迟和响应代码。为将此模块添加到 IIS 请求管道，Elastic Beanstalk 将在 IIS 配置文件 `%windir%\System32\inetsrv\config\applicationHost.config` 的 `<modules>` 部分中注册此模块。

IIS 中的 Elastic Beanstalk 模块将捕获的 Web 请求指标发送到实例上的运行状况代理（一项名为 HealthD 的 Windows 服务）。为发送此数据，此模块将使用 NetNamedPipeBinding 来提供针对计算机上通信优化的安全且可靠的绑定。

为环境配置增强型运行状况规则

AWS Elastic Beanstalk 增强型运行状况报告依靠一组规则来确定环境的运行状况。其中一些规则可能不适合您的特定应用程序。以下是一些常见的示例：

- 您使用客户端测试工具。在这种情况下，预计会频繁出现 HTTP 客户端 (4xx) 错误。
- 您将 AWS WAF 与环境的 Application Load Balancer 结合使用来阻止不需要的传入流量。在这种情况下，Application Load Balancer 为每条被拒绝的传入消息返回 HTTP 403。

默认情况下，Elastic Beanstalk 在确定环境的运行状况时会将所有应用程序 HTTP 4xx 错误纳入考虑。它会根据错误率将环境运行状况从 OK（良好）更改为 Warning（警告）、Degraded（降级）或 Severe（严重）。您可以通过 Elastic Beanstalk 配置一些增强型运行状况规则，以便正确处理诸如我们提到的示例这样的情况。您可以选择忽略环境实例上的应用程序 HTTP 4xx 错误，或者忽略环境负载均衡器返回的 HTTP 4xx 错误。本主题介绍如何进行这些配置更改。

> **Note**
>
> 目前，这些是唯一可用的增强型运行状况规则自定义设置。您无法配置增强型运行状况规则来忽略除 4xx 之外的其他 HTTP 错误。

使用 Elastic Beanstalk 控制台配置增强型运行状况规则

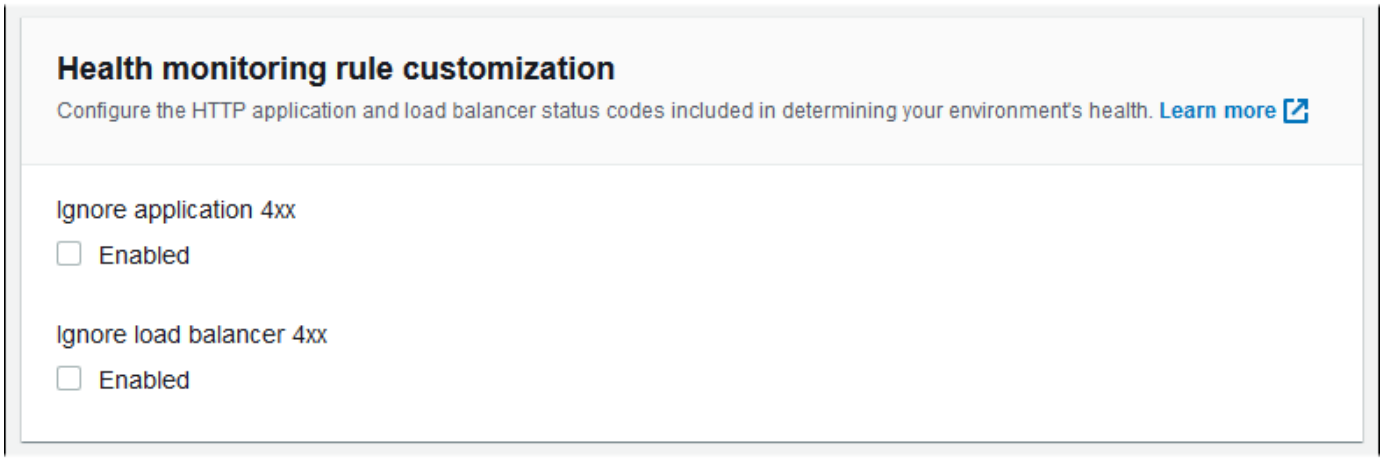
您可以使用 Elastic Beanstalk 控制台在您的环境中配置增强型运行状况规则。

使用 Elastic Beanstalk控制台配置 HTTP 4xx 状态代码检查

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Monitoring (监控) 配置类别中，选择 Edit (编辑)。
5. 在 Health monitoring rule customization (运行状况监控规则自定义) 下，启用或禁用所需的 Ignore (忽略) 选项。



6. 要保存更改，请选择页面底部的 Apply（应用）。

使用 EB CLI 配置增强型运行状况规则

您可以通过以下方法使用 EB CLI 配置增强型运行状况规则：在本地保存您的环境的配置，添加用于配置增强型运行状况规则的条目，然后将该配置上传到 Elastic Beanstalk。您可以在环境创建期间或创建后将保存的配置应用于环境。

使用 EB CLI 和保存的配置来配置 HTTP 4xx 状态代码检查

1. 使用 eb init 初始化您的项目文件夹。
2. 通过运行 eb create 命令创建环境。
3. 通过运行 eb config save 命令在本地保存配置模板。以下示例使用 `--cfg` 选项指定配置的名称。

```
$ eb config save --cfg 01-base-state
Configuration saved at: ~/project/.elasticbeanstalk/saved_configs/01-base-
state.cfg.yml
```

4. 在文本编辑器中打开保存的配置文件。
5. 在 `OptionSettings` > `aws:elasticbeanstalk:healthreporting:system:` 下，添加 `ConfigDocument` 键以列出要配置的各个增强型运行状况规则。以下 `ConfigDocument` 会禁用应用程序 HTTP 4xx 状态代码检查，同时保持启用负载均衡器 HTTP 4xx 代码检查。

```
OptionSettings:
  ...
  aws:elasticbeanstalk:healthreporting:system:
    ConfigDocument:
      Rules:
        Environment:
          Application:
            ApplicationRequests4xx:
              Enabled: false
          ELB:
            ELBRequests4xx:
              Enabled: true
      Version: 1
    SystemType: enhanced
...
```

> **Note**
>
> 您可在同一个 ConfigDocument 选项设置中结合 Rules 和 CloudWatchMetrics。发布环境的 Amazon CloudWatch 自定义指标中介绍了 CloudWatchMetrics。
> 如果您之前启用了 CloudWatchMetrics，则您使用 eb config save 命令检索的配置文件在 ConfigDocument 部分中已经具有 CloudWatchMetrics 键。请勿删除它 - 将 Rules 部分添加到相同的 ConfigDocument 选项值中。

6. 保存配置文件，关闭文本编辑器。在此示例中，更新的配置文件使用与下载的配置文件不同的名称 (02-cloudwatch-enabled.cfg.yml) 保存。这将在上传文件时创建一个单独保存的配置。您可以使用与下载的文件相同的名称覆盖现有配置而无需创建新名称。
7. 使用 eb config put 命令将更新的配置文件上传到 Elastic Beanstalk。

```
$ eb config put 02-cloudwatch-enabled
```

当将 eb config get 和 put 命令用于保存的配置时，请勿包含文件名扩展名。

8. 将保存的配置应用于正在运行的环境。

```
$ eb config --cfg 02-cloudwatch-enabled
```

--cfg 选项指定一个应用于环境的命名配置文件。可以在本地或在 Elastic Beanstalk 中保存配置文件。如果两个位置中都存在带有指定名称的配置文件，则 EB CLI 将使用本地文件。

使用配置文档配置增强型运行状况规则

用于增强型运行状况规则的配置 (config) 文档是列出要配置的规则的 JSON 文档。

以下示例配置文档会禁用应用程序 HTTP 4xx 状态代码检查，并启用负载均衡器 HTTP 4xx 状态代码检查。

```
{
  "Rules": {
    "Environment": {
      "Application": {
        "ApplicationRequests4xx": {
          "Enabled": false
        }
      },
      "ELB": {
        "ELBRequests4xx": {
          "Enabled": true
        }
      }
    }
  },
  "Version": 1
}
```

对于 AWS CLI，您将文档作为 `Value` 键的值传入选项设置参数，该参数本身是一个 JSON 对象。在这种情况下，您必须对所嵌入文档中的引号进行转义。以下命令检查配置设置是否有效。

```
$ aws elasticbeanstalk validate-configuration-settings --application-name my-app --
environment-name my-env --option-settings '[
    {
        "Namespace": "aws:elasticbeanstalk:healthreporting:system",
        "OptionName": "ConfigDocument",
        "Value": "{\"Rules\": { \"Environment\": { \"Application\":
 { \"ApplicationRequests4xx\": { \"Enabled\": false } }, \"ELB\": { \"ELBRequests4xx\":
 {\"Enabled\": true } } } }, \"Version\": 1 }"
    }
]'
```

对于采用 YAML 的 `.ebextensions` 配置文件，您可以按原样提供 JSON 文档。

```
  option_settings:
    - namespace: aws:elasticbeanstalk:healthreporting:system
      option_name: ConfigDocument
      value: {
  "Rules": {
    "Environment": {
      "Application": {
        "ApplicationRequests4xx": {
          "Enabled": false
        }
      },
      "ELB": {
        "ELBRequests4xx": {
          "Enabled": true
        }
      }
    }
  },
  "Version": 1
}
```

发布环境的 Amazon CloudWatch 自定义指标

您可以将由 AWS Elastic Beanstalk 增强型运行状况报告收集的数据作为自定义指标发布到 Amazon CloudWatch。通过将指标发布到 CloudWatch，您可以跟踪资源使用率和请求延迟随负载变化的变动情况，从而监控应用程序性能随时间推移的变化和发现潜在问题。

通过将指标发布到 CloudWatch，您还可以使它们可用于监控图表和警报。免费指标 EnvironmentHealth 在您使用增强型运行状况报告时自动启用。除 EnvironmentHealth 之外的自定义指标将产生标准 CloudWatch 费用。

要为某个环境发布 CloudWatch 自定义指标，则必须先对该环境启用增强型运行状况报告。有关说明，请参阅启用 Elastic Beanstalk 增强型运行状况报告：

主题

- 增强型运行状况报告指标
- 使用 Elastic Beanstalk 控制台配置 CloudWatch 指标
- 使用 EB CLI 配置 CloudWatch 自定义指标
- 提供自定义指标配置文件

增强型运行状况报告指标

在您的环境中启用增强型运行状况报告后，增强型运行状况报告系统将自动发布一个 CloudWatch 自定义指标，即 EnvironmentHealth。要将其他指标发布到 CloudWatch，请使用 Elastic Beanstalk 控制台、EB CLI 或 .ebextensions 为您的环境配置这些指标。

您可以将以下增强型运行状况指标从环境发布到 CloudWatch。

可用指标 – 所有平台

`EnvironmentHealth`

仅环境。除非您配置其他指标，否则这是由增强型运行状况报告系统发布的唯一的 CloudWatch 指标。环境运行状况由七种状态之一表示。在 CloudWatch 控制台中，这些状态将映射到下列值：

- 0 - 正常
- 1 - 信息
- 5 - 未知
- 10 - 无数据
- 15 - 警告
- 20 - 已降级
- 25 - 严重

`InstancesSevere`, `InstancesDegraded`, `InstancesWarning`, `InstancesInfo`, `InstancesOk`, `InstancesPending`, `InstancesUnknown`, `InstancesNoData`

仅环境。些指标表示环境中处于每种运行状况的实例的数量。`InstancesNoData` 表示未在为其接收数据的实例的数量。

`ApplicationRequestsTotal`, `ApplicationRequests5xx`, `ApplicationRequests4xx`, `ApplicationRequests3xx`, `ApplicationRequests2xx`

实例和环境。表示实例或环境完成的请求的总数量以及每种状态代码类别的请求的数量。

`ApplicationLatencyP10`, `ApplicationLatencyP50`, `ApplicationLatencyP75`, `ApplicationLatencyP85`, `ApplicationLatencyP90`, `ApplicationLatencyP95`, `ApplicationLatencyP99`, `ApplicationLatencyP99.9`

实例和环境。表示完成最快的 x% 的请求所耗费的平均时间量（以秒为单位）。

`InstanceHealth`

仅实例。表示实例的当前运行状况状态。实例运行状况由七种状态之一表示。在 CloudWatch 控制台中，这些状态将映射到下列值：

- 0 - 正常
- 1 - 信息
- 5 - 未知
- 10 - 无数据
- 15 - 警告
- 20 - 已降级
- 25 - 严重

可用指标 - Linux

`CPUIrq`, `CPUIdle`, `CPUUser`, `CPUSystem`, `CPUSoftirq`, `CPUIowait`, `CPUNice`

仅实例。表示前一分钟 CPU 在每种状态下耗费时间的百分比。

`LoadAverage1min`

仅实例。前一分钟实例的平均 CPU 负载。

`RootFilesystemUtil`

仅实例。表示正在使用的磁盘空间的百分比。

可用指标 - Windows

`CPUIdle`, `CPUUser`, `CPUPriveleged`

仅实例。表示前一分钟 CPU 在每种状态下耗费时间的百分比。

使用 Elastic Beanstalk 控制台配置 CloudWatch 指标

您可以使用 Elastic Beanstalk 控制台配置您的环境，以便将增强型运行状况报告指标发布到 CloudWatch 并使其可用于监控图表和警报。

在 Elastic Beanstalk 控制台中配置 CloudWatch 自定义指标

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。

2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Monitoring (监控) 配置类别中，选择 Edit (编辑)。
5. 在 Health reporting (运行状况报告) 下，选择要发布到 CloudWatch 的实例和环境指标。要选择多个指标，请在选择时按住 Ctrl 键。
6. 要保存更改，请选择页面底部的 Apply（应用）。

启用 CloudWatch 自定义指标可将它们添加到 Monitoring (监控) 页面上的可用指标列表。

使用 EB CLI 配置 CloudWatch 自定义指标

您可以通过以下方法使用 EB CLI 配置自定义指标：在本地保存您的环境的配置，添加用于定义要发布的指标的条目，然后将该配置上传到 Elastic Beanstalk。您可以在环境创建期间或创建后将保存的配置应用于环境。

使用 EB CLI 和保存的配置来配置 CloudWatch 自定义指标

1. 使用 eb init 初始化您的项目文件夹。
2. 通过运行 eb create 命令创建环境。
3. 通过运行 eb config save 命令在本地保存配置模板。以下示例使用 `--cfg` 选项指定配置的名称。

```
$ eb config save --cfg 01-base-state
Configuration saved at: ~/project/.elasticbeanstalk/saved_configs/01-base-
state.cfg.yml
```

4. 在文本编辑器中打开保存的配置文件。
5. 在 `OptionSettings` > `aws:elasticbeanstalk:healthreporting:system:` 下，添加一个 `ConfigDocument` 键以启用所需的每个 CloudWatch 指标。例如，以下 `ConfigDocument` 将在环境级别发布 `ApplicationRequests5xx` 和 `ApplicationRequests4xx` 指标，并在实例级别发布 `ApplicationRequestsTotal` 指标。

```
OptionSettings:
  ...
```

```
  aws:elasticbeanstalk:healthreporting:system:
    ConfigDocument:
      CloudWatchMetrics:
        Environment:
          ApplicationRequests5xx: 60
          ApplicationRequests4xx: 60
        Instance:
          ApplicationRequestsTotal: 60
      Version: 1
    SystemType: enhanced
...
```

在示例中，60 表示指标度量之间的秒数。目前，这是唯一支持的值。

> **Note**
>
> 您可在同一个 ConfigDocument 选项设置中结合 CloudWatchMetrics 和 Rules。为环境配置增强型运行状况规则中介绍了 Rules。
> 如果您以前使用 Rules 配置增强型运行状况规则，则您使用 eb config save 命令检索的配置文件在 ConfigDocument 部分中已经具有 Rules 键。请勿删除它 - 将 CloudWatchMetrics 部分添加到相同的 ConfigDocument 选项值中。

6. 保存配置文件，关闭文本编辑器。在此示例中，更新的配置文件使用与下载的配置文件不同的名称 (02-cloudwatch-enabled.cfg.yml) 保存。这将在上传文件时创建一个单独保存的配置。您可以使用与下载的文件相同的名称覆盖现有配置而无需创建新名称。
7. 使用 eb config put 命令将更新的配置文件上传到 Elastic Beanstalk。

```
$ eb config put 02-cloudwatch-enabled
```

当将 eb config get 和 put 命令用于保存的配置时，请勿包含文件扩展名。

8. 将保存的配置应用于正在运行的环境。

```
$ eb config --cfg 02-cloudwatch-enabled
```

--cfg 选项指定一个应用于环境的命名配置文件。可以在本地或在 Elastic Beanstalk 中保存配置文件。如果两个位置中都存在带有指定名称的配置文件，则 EB CLI 将使用本地文件。

提供自定义指标配置文件

Amazon CloudWatch 自定义指标的配置 (config) 文档是一个 JSON 文档，其中列出了要在环境和实例级别发布的指标。以下示例显示了一个在 Linux 上启用所有可用自定义指标的配置文件。

```
{
  "CloudWatchMetrics": {
    "Environment": {
      "ApplicationLatencyP99.9": 60,
      "InstancesSevere": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "InstancesUnknown": 60,
      "ApplicationLatencyP85": 60,
      "InstancesInfo": 60,
      "ApplicationRequests2xx": 60,
      "InstancesDegraded": 60,
      "InstancesWarning": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "InstancesNoData": 60,
      "InstancesPending": 60,
      "ApplicationLatencyP10": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "InstancesOk": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60
    },
    "Instance": {
      "ApplicationLatencyP99.9": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "ApplicationLatencyP85": 60,
      "CPUUser": 60,
      "ApplicationRequests2xx": 60,
      "CPUIdle": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "RootFilesystemUtil": 60,
```

```
      "LoadAverage1min": 60,
      "CPUIrq": 60,
      "CPUNice": 60,
      "CPUIowait": 60,
      "ApplicationLatencyP10": 60,
      "LoadAverage5min": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "CPUSystem": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60,
      "InstanceHealth": 60,
      "CPUSoftirq": 60
    }
  },
  "Version": 1
}
```

对于 AWS CLI，您将文档作为 `Value` 键的值传入选项设置参数，该参数本身是一个 JSON 对象。在这种情况下，您必须对所嵌入文档中的引号进行转义。

```
$ aws elasticbeanstalk validate-configuration-settings --application-name my-app --
environment-name my-env --option-settings '[
    {
        "Namespace": "aws:elasticbeanstalk:healthreporting:system",
        "OptionName": "ConfigDocument",
        "Value": "{\"CloudWatchMetrics\": {\"Environment\":
 {\"ApplicationLatencyP99.9\": 60,\"InstancesSevere\": 60,\"ApplicationLatencyP90\":
 60,\"ApplicationLatencyP99\": 60,\"ApplicationLatencyP95\": 60,\"InstancesUnknown
\": 60,\"ApplicationLatencyP85\": 60,\"InstancesInfo\": 60,\"ApplicationRequests2xx
\": 60,\"InstancesDegraded\": 60,\"InstancesWarning\": 60,\"ApplicationLatencyP50\":
 60,\"ApplicationRequestsTotal\": 60,\"InstancesNoData\": 60,\"InstancesPending
\": 60,\"ApplicationLatencyP10\": 60,\"ApplicationRequests5xx\": 60,
\"ApplicationLatencyP75\": 60,\"InstancesOk\": 60,\"ApplicationRequests3xx\": 60,
\"ApplicationRequests4xx\": 60},\"Instance\": {\"ApplicationLatencyP99.9\": 60,
\"ApplicationLatencyP90\": 60,\"ApplicationLatencyP99\": 60,\"ApplicationLatencyP95\":
 60,\"ApplicationLatencyP85\": 60,\"CPUUser\": 60,\"ApplicationRequests2xx\":
 60,\"CPUIdle\": 60,\"ApplicationLatencyP50\": 60,\"ApplicationRequestsTotal\":
 60,\"RootFilesystemUtil\": 60,\"LoadAverage1min\": 60,\"CPUIrq\": 60,\"CPUNice
\": 60,\"CPUIowait\": 60,\"ApplicationLatencyP10\": 60,\"LoadAverage5min\": 60,
\"ApplicationRequests5xx\": 60,\"ApplicationLatencyP75\": 60,\"CPUSystem\": 60,
\"ApplicationRequests3xx\": 60,\"ApplicationRequests4xx\": 60,\"InstanceHealth\": 60,
\"CPUSoftirq\": 60}},\"Version\": 1}"
```

```
    }
]'
```

对于采用 YAML 的 .ebextensions 配置文件，您可以按原样提供 JSON 文档。

```
  option_settings:
    - namespace: aws:elasticbeanstalk:healthreporting:system
      option_name: ConfigDocument
      value: {
  "CloudWatchMetrics": {
    "Environment": {
      "ApplicationLatencyP99.9": 60,
      "InstancesSevere": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "InstancesUnknown": 60,
      "ApplicationLatencyP85": 60,
      "InstancesInfo": 60,
      "ApplicationRequests2xx": 60,
      "InstancesDegraded": 60,
      "InstancesWarning": 60,
      "ApplicationLatencyP50": 60,
      "ApplicationRequestsTotal": 60,
      "InstancesNoData": 60,
      "InstancesPending": 60,
      "ApplicationLatencyP10": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "InstancesOk": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60
    },
    "Instance": {
      "ApplicationLatencyP99.9": 60,
      "ApplicationLatencyP90": 60,
      "ApplicationLatencyP99": 60,
      "ApplicationLatencyP95": 60,
      "ApplicationLatencyP85": 60,
      "CPUUser": 60,
      "ApplicationRequests2xx": 60,
      "CPUIdle": 60,
      "ApplicationLatencyP50": 60,
```

```
      "ApplicationRequestsTotal": 60,
      "RootFilesystemUtil": 60,
      "LoadAverage1min": 60,
      "CPUIrq": 60,
      "CPUNice": 60,
      "CPUIowait": 60,
      "ApplicationLatencyP10": 60,
      "LoadAverage5min": 60,
      "ApplicationRequests5xx": 60,
      "ApplicationLatencyP75": 60,
      "CPUSystem": 60,
      "ApplicationRequests3xx": 60,
      "ApplicationRequests4xx": 60,
      "InstanceHealth": 60,
      "CPUSoftirq": 60
    }
  },
  "Version": 1
}
```

将增强型运行状况报告与 Elastic Beanstalk API 结合使用

由于 AWS Elastic Beanstalk 增强型运行状况报告有角色和解决方案堆栈要求，因此您必须在发布增强型运行状况报告之前更新您使用的脚本和代码才能使用它。为了保持向后兼容，在您使用 Elastic Beanstalk API 创建环境时默认不会启用增强型运行状况报告。

您可以通过为您的环境设置服务角色、实例配置文件和 Amazon CloudWatch 配置选项来配置增强型运行状况报告。您可以用三种方式完成此操作：在 `.ebextensions` 文件夹中设置配置选项、使用保存的配置或在 `create-environment` 调用的 `option-settings` 参数中直接配置它们。

要使用 API、开发工具包或 AWS Command Line Interface (CLI) 创建支持增强型运行状况的环境，您必须：

- 创建具有适当权限的服务角色和实例配置文件
- 使用新的平台版本创建新环境
- 设置运行状况系统类型、实例配置文件和服务角色配置选项

使用 `aws:elasticbeanstalk:healthreporting:system`、`aws:autoscaling:launchconfiguration`

和 aws:elasticbeanstalk:environment 命名空间中的以下配置选项配置适用于增强型运行状况报告的环境。

增强型运行状况配置选项

SystemType

命名空间: aws:elasticbeanstalk:healthreporting:system

要启用增强型运行状况报告，请设置为 **enhanced**。

IamInstanceProfile

命名空间: aws:autoscaling:launchconfiguration

设置为配置为与 Elastic Beanstalk 一起使用的实例配置文件的名称。

ServiceRole

命名空间: aws:elasticbeanstalk:environment

设置为配置为与 Elastic Beanstalk 一起使用的服务角色的名称。

ConfigDocument（可选）

命名空间: aws:elasticbeanstalk:healthreporting:system

一个用于定义要发布到 CloudWatch 的实例和环境指标的 JSON 文档。例如：

```
{
  "CloudWatchMetrics":
    {
    "Environment":
      {
      "ApplicationLatencyP99.9":60,
      "InstancesSevere":60
      }
    "Instance":
      {
      "ApplicationLatencyP85":60,
      "CPUUser": 60
      }
    }
  "Version":1
}
```

> **Note**
>
> 可能需要对配置文件进行特殊的格式化，如转义引号，具体视您将它们提供给 Elastic Beanstalk 的方式而定。有关示例，请参阅 提供自定义指标配置文件。

增强型运行状况日志格式

AWS Elastic Beanstalk 平台使用自定义 Web 服务器日志格式将有关 HTTP 请求的信息高效地中继到增强型运行状况报告系统。系统将分析日志、确定问题并相应地设置实例和环境运行状况。如果您对环境禁用 Web 服务器代理并从 Web 容器直接响应请求，则仍然可以通过将服务器配置为以 Elastic Beanstalk 运行状况代理使用的位置和格式来输出日志，从而充分利用增强型运行状况报告。

> **Note**
>
> 此页面上的信息仅与基于 Linux 的平台相关。在 Windows Server 平台上，Elastic Beanstalk 将直接从 IIS Web 服务器接收有关 HTTP 请求的信息。有关详细信息，请参阅Windows Server 上的 IIS 中的 Web 服务器指标捕获。

Web 服务器日志配置

Elastic Beanstalk 平台配置为输出两个包含有关 HTTP 请求的信息的日志。第一个日志采用详细格式并提供关于请求的详细信息，如请求者的用户代理信息和人类可读的时间戳。

/var/log/nginx/access.log

以下示例来自在 Ruby Web 服务器环境上运行的 Nginx 代理，但格式与 Apache 的类似。

```
172.31.24.3 - - [23/Jul/2015:00:21:20 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:21 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:22 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
```

```
172.31.24.3 - - [23/Jul/2015:00:21:22 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
172.31.24.3 - - [23/Jul/2015:00:21:22 +0000] "GET / HTTP/1.1" 200 11 "-" "curl/7.22.0
 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
 librtmp/2.3" "177.72.242.17"
```

第二个日志采用简洁格式。它只包含与增强型运行状况报告相关的信息。此日志输出到一个名为 `healthd` 的子文件夹，并且每小时轮换一次。旧日志在轮换出之后将会立即被删除。

/var/log/nginx/healthd/application.log.2015-07-23-00

以下示例显示了一个机器可读格式的日志。

```
1437609879.311"/"200"0.083"0.083"177.72.242.17
1437609879.874"/"200"0.347"0.347"177.72.242.17
1437609880.006"/bad/path"404"0.001"0.001"177.72.242.17
1437609880.058"/"200"0.530"0.530"177.72.242.17
1437609880.928"/bad/path"404"0.001"0.001"177.72.242.17
```

增强型运行状况日志格式包括以下信息：

- 请求的时间，用 Unix 时间表示
- 请求的路径
- 结果的 HTTP 状态代码
- 请求时间
- 上游时间
- `X-Forwarded-For` HTTP 标头

对于 nginx 代理，时间印在浮点秒中，有三个小数位。对于 Apache，将用整微秒表示。

> **Note**
>
> 如果您在日志文件中看到类似于下面的警告，其中 `DATE-TIME` 是日期和时间，并且您使用了自定义代理，例如在多容器 Docker 环境中，您必须使用 .ebextension 来配置环境，这样 `healthd` 才能读取您的日志文件：
>
> ```
> W, [DATE-TIME #1922] WARN -- : log file "/var/log/nginx/healthd/
> application.log.DATE-TIME" does not exist
> ```

您可在多容器 Docker 示例中开始使用 .ebextension。

/etc/nginx/conf.d/webapp_healthd.conf

以下示例显示了 Nginx 的日志配置，并突出显示了 healthd 日志格式。

```
upstream my_app {
  server unix:///var/run/puma/my_app.sock;
}

log_format healthd '$msec"$uri"'
                '$status"$request_time"$upstream_response_time"'
                '$http_x_forwarded_for';

server {
  listen 80;
  server_name _ localhost; # need to listen to localhost for worker tier

  if ($time_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {
    set $year $1;
    set $month $2;
    set $day $3;
    set $hour $4;
  }

  access_log  /var/log/nginx/access.log  main;
  access_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour healthd;

  location / {
    proxy_pass http://my_app; # match the name of upstream directive which is defined
 above
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  }

  location /assets {
    alias /var/app/current/public/assets;
    gzip_static on;
    gzip on;
    expires max;
    add_header Cache-Control public;
  }
```

```
  location /public {
    alias /var/app/current/public;
    gzip_static on;
    gzip on;
    expires max;
    add_header Cache-Control public;
  }
}
```

/etc/httpd/conf.d/healthd.conf

以下示例显示了 Apache 的日志配置。

```
LogFormat "%{%s}t\"%U\"%s\"%D\"%D\"%{X-Forwarded-For}i" healthd
CustomLog "|/usr/sbin/rotatelogs /var/log/httpd/healthd/application.log.%Y-%m-%d-%H
 3600" healthd
```

为增强型运行状况报告生成日志

要将日志提供给运行状况代理，您必须执行以下操作：

- 以正确格式输出日志，如前一部分所示
- 将日志输出到 `/var/log/nginx/healthd/`
- 使用以下格式命名日志：`application.log.$year-$month-$day-$hour`
- 每小时将日志轮换一次
- 请勿截断日志

通知和问题排查

本页列出了常见问题的示例原因消息和指向更多信息的链接。原因消息显示在 Elastic Beanstalk 控制台的环境概述页面中，当运行状况问题在多次检查中持续出现时，原因消息将记录在事件中。

部署

Elastic Beanstalk 监控您的环境以确保一致的后续部署。如果滚动部署失败，则在您的环境中的实例上运行的应用程序版本可能会不同。如果部署在一个或多个批次上成功，但在所有批次完成前失败，则可能出现这种情况。

在总共 5 个实例的 2 个实例上发现不正确的应用程序版本。预期版本“v1”(deployment 1)。

在环境实例上发现了不正确的应用程序版本。预期版本“v1”(deployment 1)。

环境中部分或所有实例上没有运行预期的应用程序版本。

不正确的应用程序版本“v2”(deployment 2)。预期版本“v1”(deployment 1)。

部署到实例的应用程序与预期版本不同。如果部署失败，则预期版本重置为最近成功部署的版本。在以上示例中，第一个部署 (版本“V1”) 成功，但第二个部署 (版本“V2”) 失败。任何运行“V2”的实例将被视为运行状况不佳。

要解决此问题，请启动另一个部署。您可以重新部署以前已知可以正常工作的版本，或者配置您的环境在部署和重新部署新版本期间忽略运行状况检查，以强制部署完成。

您还可以确定和终止运行错误应用程序版本的实例。Elastic Beanstalk 将使用正确版本启动实例来替换任何您终止的实例。使用 EB CLI 运行状况命令来确定运行错误应用程序版本的实例。

应用程序服务器

15% 的 HTTP 请求出错，错误消息为 HTTP 4xx

20% 的 ELB 请求出现 HTTP 4xx 错误。

对实例或环境的 HTTP 请求中有很大一部分失败，错误消息为 4xx。

400 系列状态代码表示用户发出了一个错误请求，如请求了不存在的页面 (404 未找到文件) 或用户无法访问 (403 禁止访问)。少量的 404 是正常的，但如果数量多，则可能表示存在指向不可用的页面的内部或外部链接。可以通过修复错误的内部链接并为错误的外部链接添加重定向来解决这些问题。

5% 的请求失败，错误消息为 HTTP 5xx

3% 的 ELB 请求失败，错误消息为 HTTP 5xx。

对实例或环境的 HTTP 请求中有很大一部分失败，错误消息为 500 系列状态代码。

500 系列状态代码表示应用程序服务器遇到了内部错误。这些问题表示应用程序代码中存在错误，应该快速加以确定和修复。

95% 的 CPU 正在使用中

对于一个实例，运行状况代理将会报告极高百分比的 CPU 使用率并将实例运行状况设置为警告或降级。

请扩展您的环境以减少实例的负载。

工作线程实例

20 条消息正在队列中等待（25 秒前）

向您的工作线程环境的队列添加请求的速度超过了处理这些请求的速度。请扩展您的环境以增加容量。

5 条消息位于死信队列中（15 分钟前）

工作线程请求重复失败，正在添加到the section called “死信队列”。请检查死信队列中的请求以了解它们失败的原因。

其他资源

4 个活动实例小于 Auto Scaling 组的最小大小 5

在您的环境中运行的实例数少于为 Auto Scaling 组配置的最小数量。

Auto Scaling 组 (groupname) 通知已删除或修改

为 Auto Scaling 组配置的通知已在 Elastic Beanstalk 外部修改。

管理警报

您可以使用 Elastic Beanstalk 控制台为监控的指标创建警报。警报可以帮助您监控对 AWS Elastic Beanstalk 环境进行的更改，以使您轻松确定问题并在出现之前加以缓解。例如，您可以设置一个警报以在环境中的 CPU 使用率超过特定阈值时通知您，从而确保在潜在问题出现之前您就得到通知。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon CloudWatch 结合使用。

> **Note**
>
> Elastic Beanstalk 将 CloudWatch 用于监控和警报，这意味着您所使用的任何警报都会对您的AWS账户产生 CloudWatch 费用。

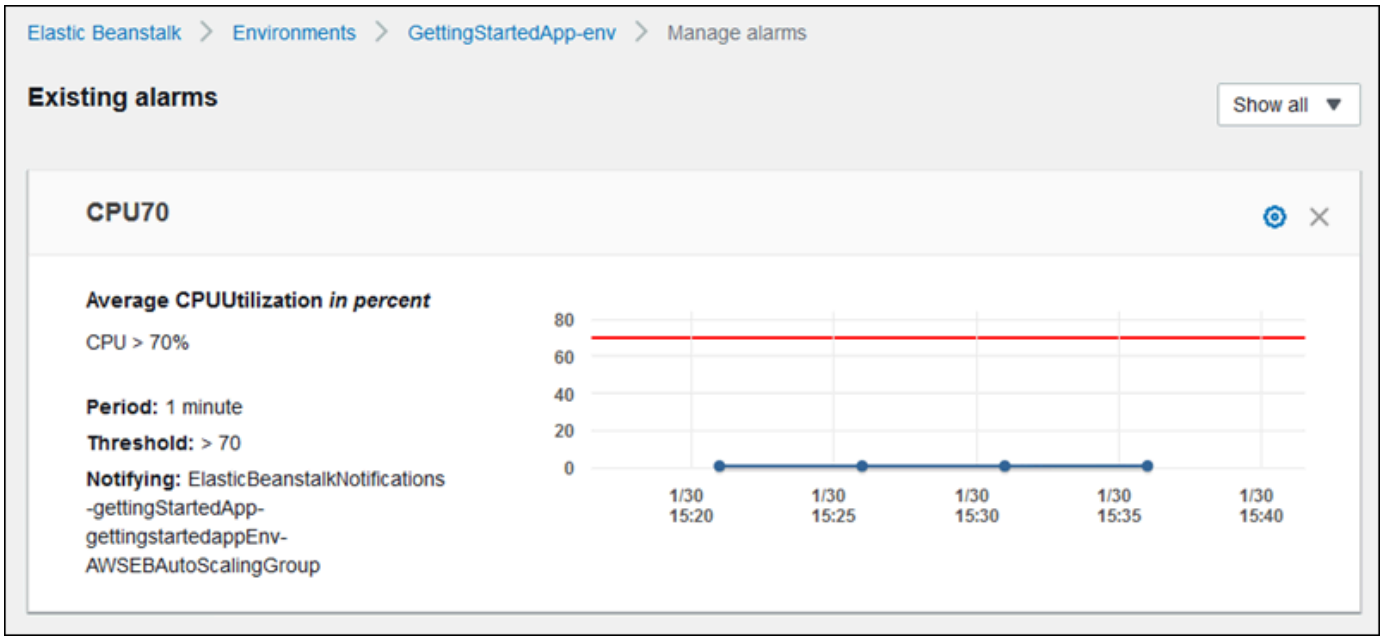
有关特定于监控的指标的更多信息，请参阅基本运行状况报告。

检查警报的状态

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Alarms (告警)。



该页面显示现有警报的列表。如果有任何警报处于警报状态，则它们会标记有 （警告）。

4. 要筛选警报，请选择下拉菜单，然后选择筛选器。
5. 要编辑或删除警报，请分别选择 （编辑）或 （删除）。

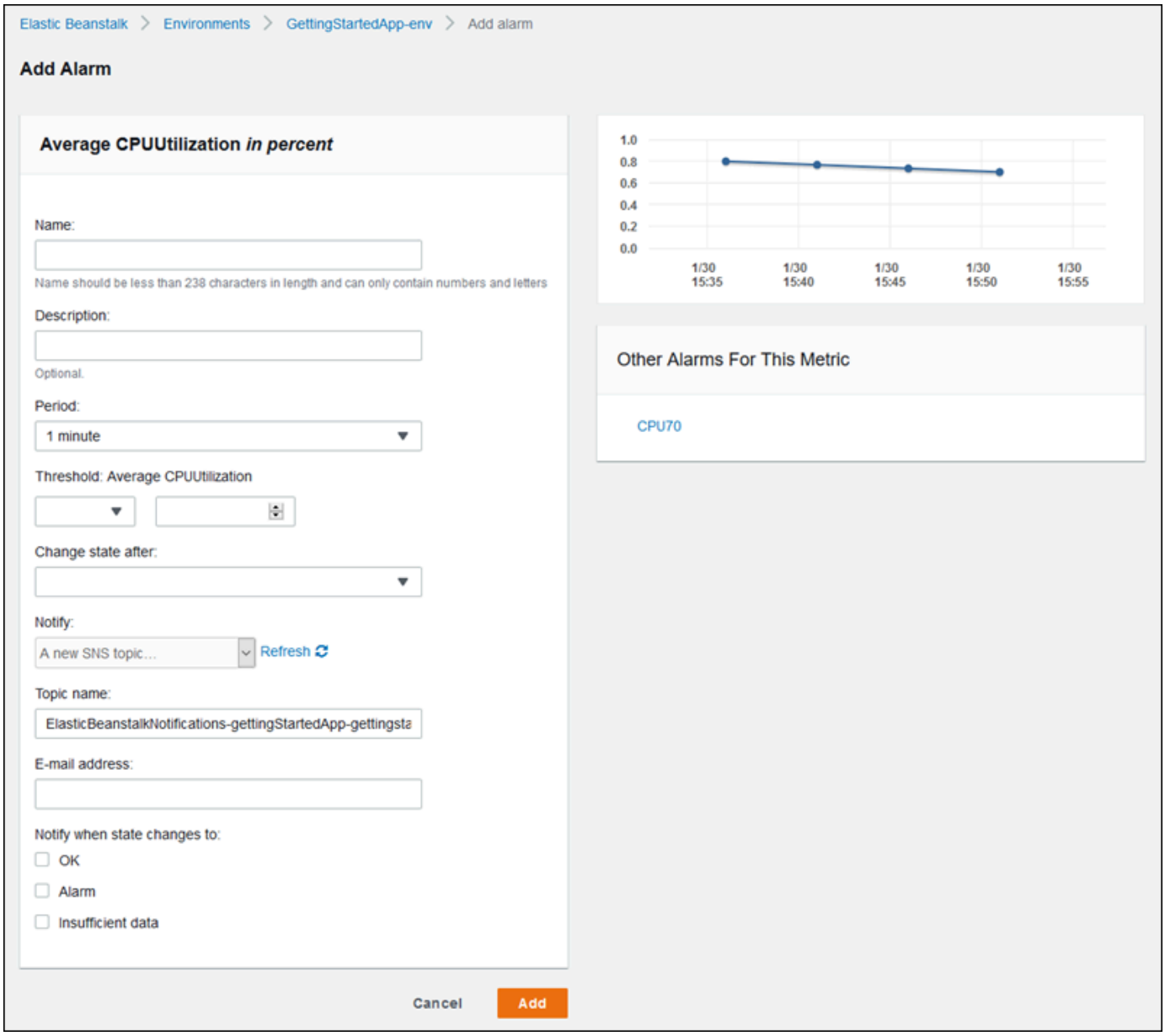
创建警报

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Monitoring (监控)。
4. 找到要为其创建警报的指标，然后选择

   （警报）。将显示 Add alarm (添加警报) 页面。



5. 输入有关警报的详细信息：

- Name (名称)：此警报的名称。
- Description (描述)（可选）：有关此警报含义的简短描述。
- Period (周期)：两次读取之间的时间间隔。
- Threshold (阈值)：描述为触发警报而必须使该指标超过的值和逾越的行为。
- Change state after (更改状态的时间)：超过阈值之后触发警报状态更改所需的时长。
- Notify (通知)：警报更改状态时通知的 Amazon SNS 主题。
- 更改为如下状态时通知：
  - OK (正常)：指标在规定的阈值范围内。
  - Alarm (警报)：指标超过规定的阈值。
  - Insufficient data (数据不足)：警报刚刚开始、指标不可用或没有足够数据可用于指标来确定警报状态。

6. 选择 Add（添加）。环境更新时，环境状态会更改为灰色。您可以通过在导航窗格中选择 Alarms (警报) 来查看您创建的警报。

查看 Elastic Beanstalk 环境的更改历史记录

您可以使用 AWS 管理控制台查看对 Elastic Beanstalk 环境进行的配置更改的历史记录。Elastic Beanstalk 从 AWS CloudTrail 中记录的事件中获取您的更改历史记录，并将其显示在一个列表中，以便您可以轻松浏览和筛选。

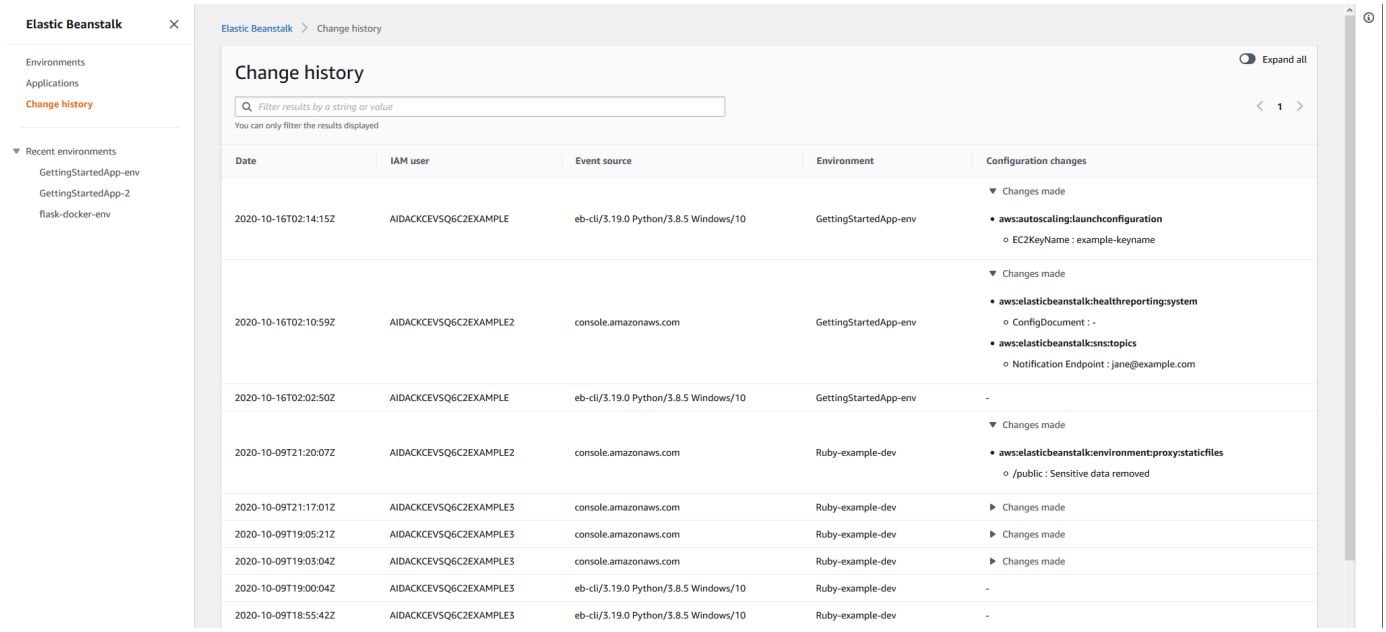
“更改历史记录”面板显示对环境所做更改的以下信息：

- 进行更改的日期和时间
- 负责所做更改的 IAM 用户
- 用于进行更改的源工具（Elastic Beanstalk 命令行界面 (EB CLI) 或控制台）
- 配置参数和设置的新值

属于更改一部分的任何敏感数据（例如受更改影响的数据库用户的姓名）都不会显示在面板中。

要查看更改历史记录

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Change history（更改历史记录）。



Change History（更改历史记录）页面显示了对 Elastic Beanstalk 环境所做配置更改的列表。您可以通过选择 **<**（上一步）或 **>**（下一步）或者选择特定的页码来浏览列表。在 Configuration changes（配置更改）列下，选择箭头图标以在 Changes made（所做更改）标题下的展开和折叠更改列表之间切换。使用搜索栏从更改历史记录列表中筛选结果。您可以输入任何字符串来缩小显示的更改列表的范围。

请注意有关筛选显示的结果的以下事项：

- 搜索筛选条件不区分大小写。
- 您可以根据 Configuration changes（配置更改）列下的信息筛选显示的更改，即使它因为折叠在 Changes made（所做更改）内而不可见。
- 您只能筛选显示的结果。但是，即使您选择转到另一个页面以显示更多结果，筛选条件仍然保持不变。您筛选的结果还将附加到下一页的结果集中。

以下示例演示了如何筛选之前的屏幕上显示的数据：

- 在搜索框中输入 **`GettingStartedApp-env`** 以缩小结果范围，使其仅显示对名称 GettingStartedApp-env 的环境所做的更改。
- 在搜索框中输入 **`example3`** 以缩小结果的范围，使其仅包含用户名中包含字符串 example3 的 IAM 用户所做的更改。

- 在搜索框中输入 **2020-10** 以缩小结果范围，使其仅包含在 2020 年 10 月当月所做的更改。将搜索值更改为 **2020-10-16** 以进一步筛选显示的结果，使其仅包含 2020 年 10 月 16 日当天所做的更改。
- 在搜索框中输入 **proxy:staticfiles** 以缩小结果的范围，使其仅包含对名为 aws:elasticbeanstalk:environment:proxy:staticfiles 的命名空间所做的更改。显示的行是筛选器的结果。即使对于折叠在 Changes made（所做更改）下的结果也是如此。

查看 Elastic Beanstalk 环境的事件流

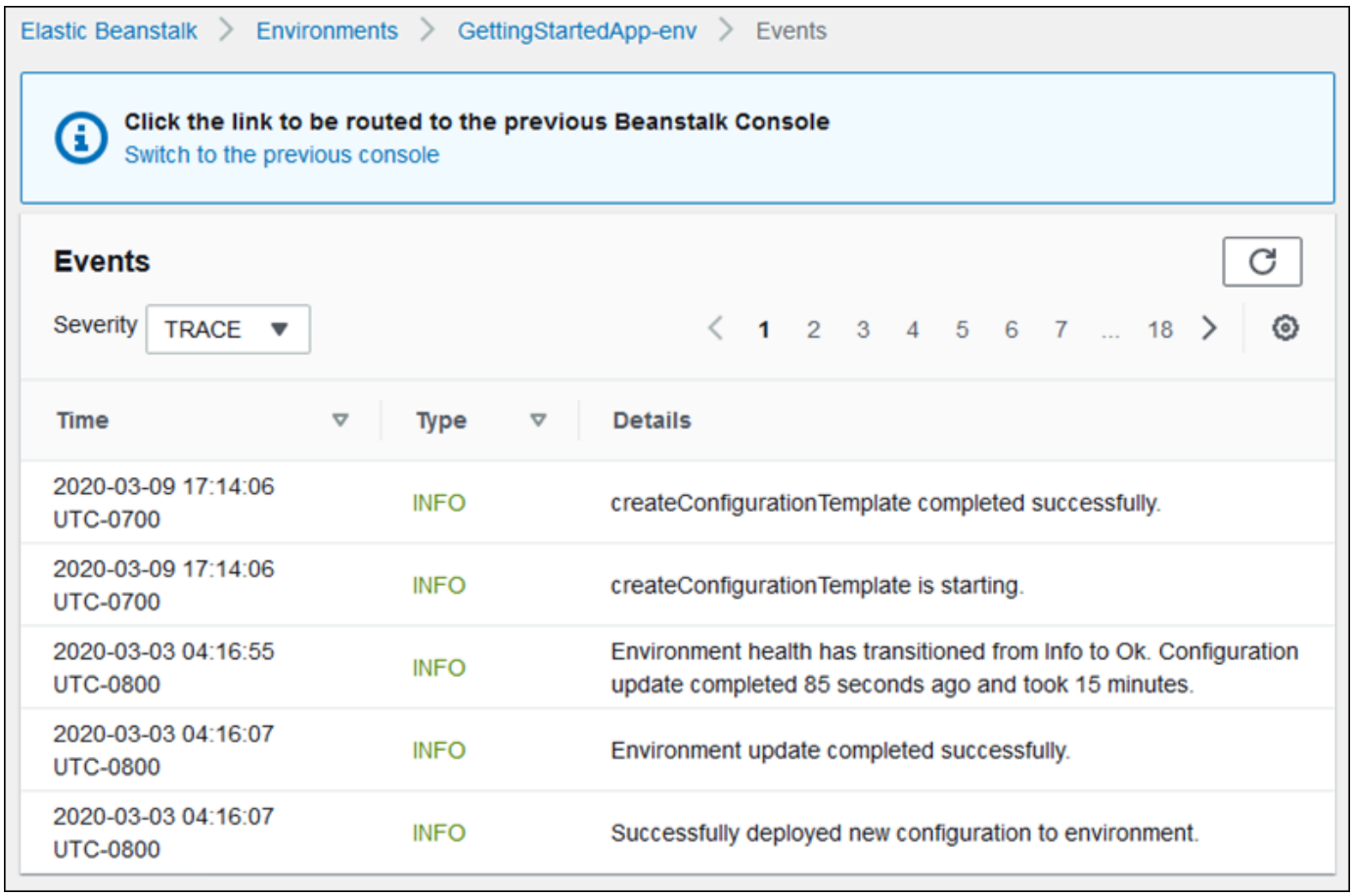
您可以使用 AWS 管理控制台访问与应用程序相关联的事件和通知。

查看事件

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Events (事件)。



“Events (事件)”页面显示已为环境记录的所有事件的列表。可以通过选择 < (上一页) 、> (下一页) 或页码来分页查看列表。您可以使用严重性下拉列表按显示的事件类型进行筛选。

EB CLI 和 AWS CLI 均提供检索事件的命令。如果您正在使用 EB CLI 管理您的环境，请使用 eb events 打印事件的列表。此命令还拥有一个 `--follow` 选项，此选项可让系统继续显示新事件，直到您按 Ctrl+C 停止输出。

要使用 AWS CLI 拉取事件，请使用 `describe-events` 命令并通过名称或 ID 指定环境：

```
$ aws elasticbeanstalk describe-events --environment-id e-gbjzqccra3
{
    "Events": [
        {
            "ApplicationName": "elastic-beanstalk-example",
            "EnvironmentName": "elasticBeanstalkExa-env",
            "Severity": "INFO",
            "RequestId": "a4c7bfd6-2043-11e5-91e2-9114455c358a",
            "Message": "Environment update completed successfully.",
```

```
            "EventDate": "2015-07-01T22:52:12.639Z"
        },
...
```

有关命令行工具的更多信息，请参阅工具。

列出和连接到服务器实例

您可以通过 Elastic Beanstalk 控制台查看运行您的 AWS Elastic Beanstalk 应用程序环境的 Amazon EC2 实例列表。您可以使用任何 SSH 客户端连接到这些实例。您可使用远程桌面连接运行 Windows 的实例。

有关特定开发环境的一些说明：

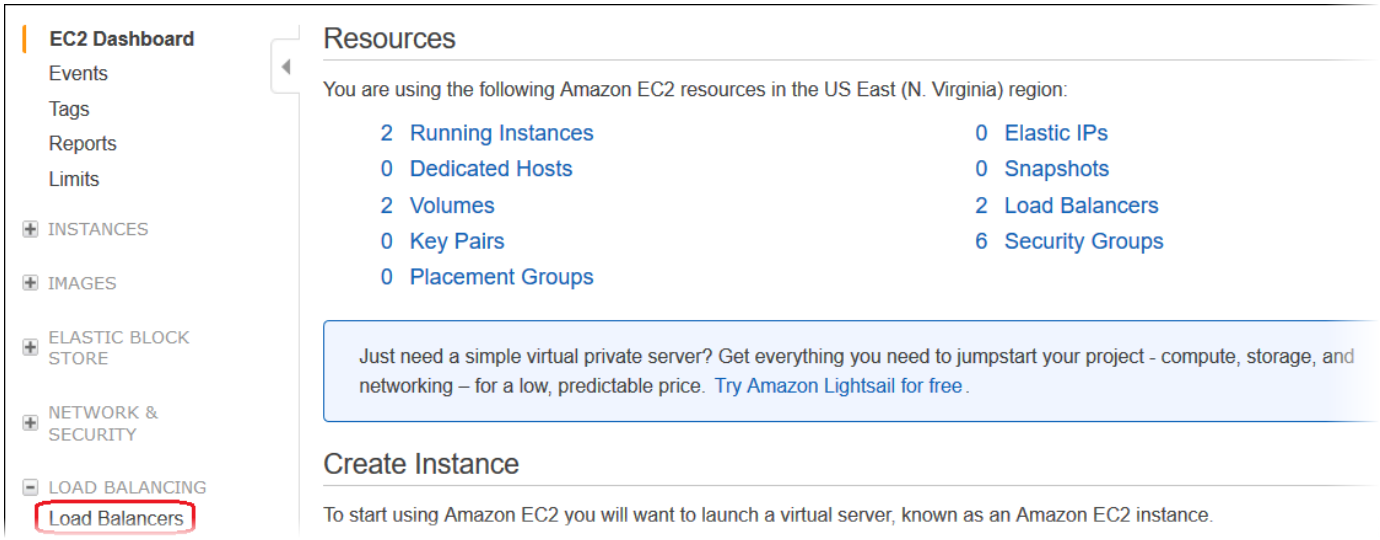
- 有关使用列出和连接服务器实例的更多信息 AWS Toolkit for Eclipse，请参阅列出和连接到服务器实例。
- 有关使用列出和连接服务器实例的更多信息 AWS Toolkit for Visual Studio，请参阅列出和连接到服务器实例。

> **Important**
>
> 您必须创建 Amazon EC2 密钥对并将 Elastic Beanstalk 预配置的 Amazon EC2 实例配置为使用 Amazon EC2 密钥对，然后才能访问 Elastic Beanstalk 预配置的 Amazon EC2 实例。您可使用 AWS 管理控制台设置 Amazon EC2 密钥对。有关创建 Amazon EC2 密钥对的说明，请参阅 Amazon EC2 入门指南。有关如何配置 Amazon EC2 实例以使用 Amazon EC2 密钥对的详细信息，请参阅 EC2 密钥对。
>
> 默认情况下，Elastic Beanstalk 不启用与 Windows 容器中的 EC2 实例的远程连接（早期 Windows 容器中的 EC2 实例除外）。（Elastic Beanstalk 将早期 Windows 容器中的 EC2 实例配置为使用端口 3389 进行 RDP 连接。）您可以向安全组添加授权入站流量进入实例的规则，进而启用与您运行 Windows 的 EC2 实例的远程连接。我们强烈建议您在结束远程连接时删除该规则。您可以在下次需要远程登录时再次添加该规则。有关更多信息，请参阅《适用于微软 Windows 的亚马逊弹性计算云用户指南》中的为 Windows 实例的入站 RDP 流量添加规则并连接到您的 Windows 实例。

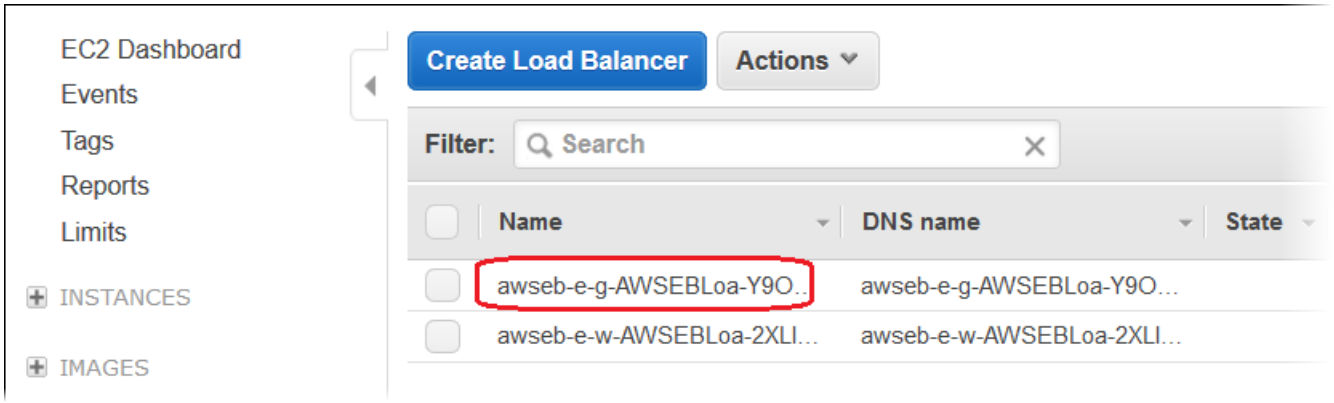
查看和连接到环境的 Amazon EC2 实例

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

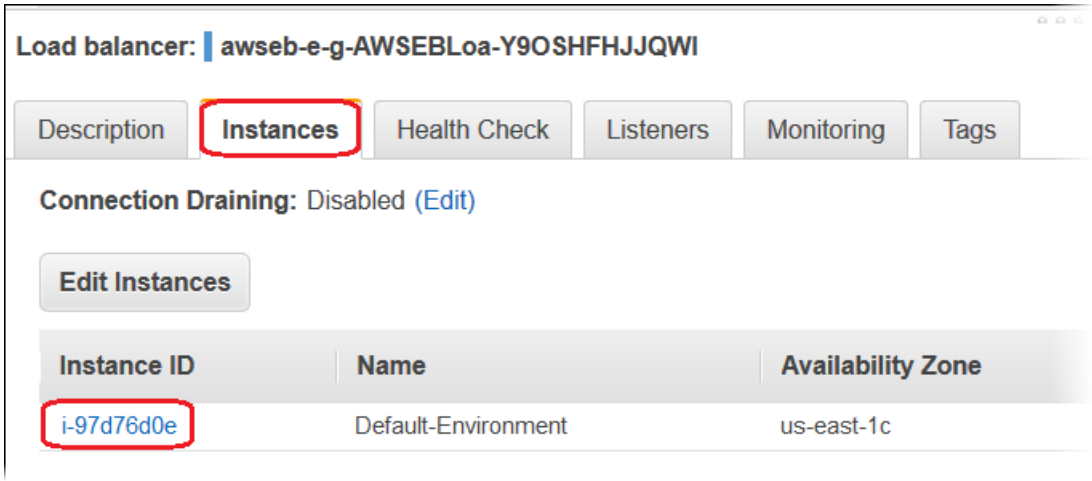
2. 在控制台的导航窗格中，选择 Load Balancers (负载均衡器)。



3. 由 Elastic Beanstalk 创建的负载均衡器的名称中有 awseb。查找适合您环境的负载均衡器，然后单击该均衡器。

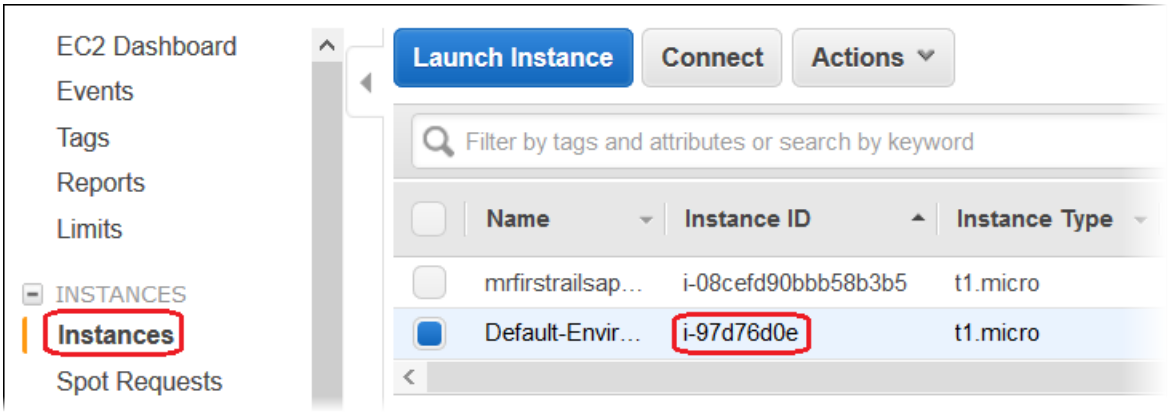


4. 选择控制台底部窗格中的 Instances (实例) 选项卡。



显示 Elastic Beanstalk 环境的负载均衡器所使用的实例列表。请记下您想连接的实例 ID。

5. 在 Amazon EC2 控制台的导航窗格中，选择 Instances (实例)，然后在列表中找到您的实例 ID。



6. 右键单击在环境负载均衡器中运行的 Amazon EC2 实例的实例 ID，然后从上下文菜单中选择 Connect (连接)。
7. 请记下该实例在 Description (描述) 选项卡上的公共 DNS 地址。
8. 使用你选择的 SSH 客户端连接到运行 Linux 的实例，然后键入 ssh -i .ec2/mykeypair.pem ec2-user@<public- DNS->。of-the-instance

有关连接亚马逊 EC2 Linux 实例的更多信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 Linux 实例入门。

如果您的 Elastic Beanstalk 环境在 Windows 服务器平台上使用.NET，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 Windows 实例入门。

查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志

Elastic Beanstalk 环境中的 Amazon EC2 实例会生成日志，您可以查看这些日志来对应用程序或配置文件进行故障排除。由 Web 服务器、应用程序服务器、Elastic Beanstalk 平台脚本和 AWS CloudFormation 创建的日志在本地存储在各个实例上。您可使用环境管理控制台或 EB CLI 轻松检索这些日志。您还可以将环境配置为将日志实时流式传输到 Amazon CloudWatch Logs。

结尾日志是最常用日志文件的后 100 行，最常用日志文件有 Elastic Beanstalk 操作日志和来自 Web 服务器或应用程序服务器的日志。当您使用环境管理控制台或 eb logs 请求结尾日志时，环境中的一个实例会将最新的日志条目连接为一个文本文件，并将其上传到 Amazon S3。

捆绑日志是包含更多日志文件的完整日志，包括来自 yum 和 cron 的日志以及一些来自 AWS CloudFormation 的日志。当您请求捆绑日志时，环境中的一个实例会将完整日志文件打包为一个 ZIP 存档并上传到 Amazon S3。

> **Note**
>
> Elastic Beanstalk Windows Server 平台不支持捆绑日志。

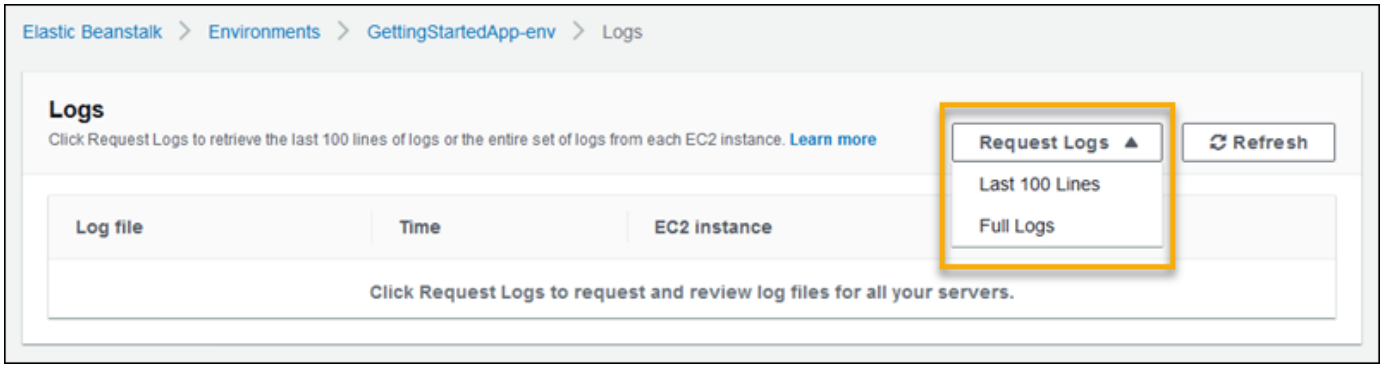
要将已轮换的日志上传到 Amazon S3，环境中的实例必须有一个实例配置文件，并且具有向您的 Elastic Beanstalk Amazon S3 存储桶写入的权限。这些权限包含在默认实例配置文件中，在 Elastic Beanstalk 控制台中首次启动环境时，Elastic Beanstalk 将提示您创建该配置文件。

检索实例日志

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择日志。
4. 选择 Request Logs (请求日志)，然后选择要检索的日志类型。要检索结尾日志，请选择 Last 100 Lines (最后 100 行)。要检索捆绑日志，请选择 Full Logs (完整日志)。



5. Elastic Beanstalk 完成检索日志后，选择 Download (下载)。

Elastic Beanstalk 将结尾和捆绑日志存储在 Amazon S3 存储桶中，并生成可用于访问日志的预签名 Amazon S3 URL。Elastic Beanstalk 会在 15 分钟的持续时间之后，从 Amazon S3 中删除文件。

> **Warning**
>
> 拥有该预签名 Amazon S3 URL 的任何人都可以在这些文件被删除之前访问这些文件。只能将该 URL 提供给可信方。

> **Note**
>
> 您的用户策略必须具有 `s3:DeleteObject` 权限。Elastic Beanstalk 使用您的用户权限从 Amazon S3 中删除日志。

要保留日志，您可以将环境配置为在日志轮换后自动将它们发布到 Amazon S3。要启用到 Amazon S3 的日志轮换，请按照配置实例日志查看中的过程操作。环境中的实例会尝试每小时上传一次已轮换的日志。

如果应用程序不是在环境平台默认配置的位置生成日志，可以使用配置文件 (`.ebextensions`) 扩展默认配置。您可以将应用程序的日志文件添加到结尾日志、捆绑日志或日志轮换中。

要实现实时日志流和长期存储，请将您的环境配置为将日志流式传输到 Amazon CloudWatch Logs。

小节目录

Amazon EC2 实例上的日志位置

日志存储在您环境中 Amazon EC2 实例上的标准位置。Elastic Beanstalk 生成以下日志。

Amazon Linux 2

- `/var/log/eb-engine.log`

Amazon Linux AMI (AL1)

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

- `/var/log/eb-activity.log`
- `/var/log/eb-commandprocessor.log`

Windows Server

- `C:\Program Files\Amazon\ElasticBeanstalk\logs\`
- `C:\cfn\log\cfn-init.log`

这些日志中包含有关部署活动的消息，包括与配置文件（`.ebextensions`）有关的消息。

各应用程序和 Web 服务器都在其自己的文件夹中存储日志：

- Apache – `/var/log/httpd/`
- IIS – `C:\inetpub\wwwroot\`
- Node.js – `/var/log/nodejs/`
- nginx – `/var/log/nginx/`
- Passenger – `/var/app/support/logs/`
- Puma – `/var/log/puma/`
- Python – `/opt/python/log/`
- Tomcat – `/var/log/tomcat/`

Amazon S3 中的日志位置

当您从环境请求结尾日志或捆绑日志时，或在实例上传轮换日志时，它们存储在 Amazon S3 中您的 Elastic Beanstalk 存储桶中。Elastic Beanstalk 为您在其中创建环境的每个AWS区域创建一个名为 `elasticbeanstalk-`*`region`*`-`*`account-id`* 的存储桶。在此存储桶内，日志存储在路径 `resources/environments/logs/`*`logtype`*`/`*`environment-id`*`/`*`instance-id`* 下。

例如，Elastic Beanstalk 环境 e-mpcwnwheky（位于账户 123456789012 的AWS区域 us-west-2 中）的实例 i-0a1fd158 的日志存储在以下位置：

- 结尾日志 –

  s3://elasticbeanstalk-us-west-2-123456789012/resources/environments/logs/tail/e-mpcwnwheky/i-0a1fd158

- 捆绑日志 –

  s3://elasticbeanstalk-us-west-2-123456789012/resources/environments/logs/bundle/e-mpcwnwheky/i-0a1fd158

- 轮换日志 –

  s3://elasticbeanstalk-us-west-2-123456789012/resources/environments/logs/publish/e-mpcwnwheky/i-0a1fd158

> **Note**
>
> 您可以在环境管理控制台中查找您的环境 ID。

Elastic Beanstalk 在结尾日志和捆绑日志创建 15 分钟后自动将它们从 Amazon S3 中删除。轮换日志会保留到您将它们删除或移入 S3 Glacier。

Linux 上的日志轮换设置

在 Linux 平台上，Elastic Beanstalk 使用 logrotate 来定期轮换日志。如果已配置，则日志在本地轮换后，将由日志轮换任务选取并上传至 Amazon S3。默认情况下，已本地轮换的日志不会显示在结尾日志或捆绑日志中。

您可在 /etc/logrotate.elasticbeanstalk.hourly/ 中找到 logrotate 的 Elastic Beanstalk 配置文件。这些轮换设置是平台特定的，以后的平台版本中可能会有变化。如需有关可用的设置和示例配置的更多信息，请运行 man logrotate。

配置文件在 /etc/cron.hourly/ 中由 cron 任务调用。有关 cron 的更多信息，请运行 man cron。

扩展默认日志任务配置

Elastic Beanstalk 使用 Amazon EC2 实例上的子文件夹 `/opt/elasticbeanstalk/tasks` (Linux) 或 `C:\Program Files\Amazon\ElasticBeanstalk\config` (Windows Server) 中的文件配置结尾日志、捆绑日志和日志轮换任务。

在 Amazon Linux 上：

- 结尾日志 –

  ```
  /opt/elasticbeanstalk/tasks/taillogs.d/
  ```
- 捆绑日志 –

  ```
  /opt/elasticbeanstalk/tasks/bundlelogs.d/
  ```
- 轮换日志 –

  ```
  /opt/elasticbeanstalk/tasks/publishlogs.d/
  ```

在 Windows Server 上：

- 结尾日志 –

  ```
  c:\Program Files\Amazon\ElasticBeanstalk\config\taillogs.d\
  ```
- 轮换日志 –

  ```
  c:\Program Files\Amazon\ElasticBeanstalk\config\publogs.d\
  ```

例如，Linux 上的 `eb-activity.conf` 文件将两个日志文件添加到结尾日志任务。

`/opt/elasticbeanstalk/tasks/taillogs.d/eb-activity.conf`

```
/var/log/eb-commandprocessor.log
/var/log/eb-activity.log
```

您可以使用环境配置文件 (`.ebextensions`) 将自己的 `.conf` 文件添加到这些文件夹。`.conf` 文件列出应用程序特定的日志文件，Elastic Beanstalk 将这些日志文件添加到日志文件任务。

使用 files 部分可将配置文件添加到要修改的任务。例如，以下配置文本向您的环境中的每个实例添加一个日志配置文件。此日志配置文件 (即 cloud-init.conf) 向结尾日志添加 /var/log/cloud-init.log。

```
files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/cloud-init.conf" :
    mode: "000755"
    owner: root
    group: root
    content: |
      /var/log/cloud-init.log
```

将此文本添加到扩展名为 .config 的文件中，并将该文件添加到您的源包中名为 .ebextensions 的文件夹下。

```
~/workspace/my-app
|-- .ebextensions
|   `-- tail-logs.config
|-- index.php
`-- styles.css
```

在 Linux 平台上，您也可以在日志任务配置中使用通配符。此配置文件将应用程序根目录下 .log 文件夹中扩展名为 log 的所有文件添加到捆绑日志中。

```
files:
  "/opt/elasticbeanstalk/tasks/bundlelogs.d/applogs.conf" :
    mode: "000755"
    owner: root
    group: root
    content: |
      /var/app/current/log/*.log
```

在 Windows 平台上，日志任务配置不支持通配符。

> **Note**
>
> 为了帮助您自行熟悉日志自定义过程，您可以使用 EB CLI 部署示例应用程序。为此，EB CLI 创建一个本地应用程序目录，其中包含 .ebextentions 子目录以及示例配置。还可以使用示例应用程序的日志文件来探索本主题中描述的日志检索功能。有关如何使用 EB CLI 创建示例应用程序的更多信息，请参阅 EB CLI 基础知识。

有关使用配置文件的更多信息，请参阅使用配置文件 (.ebextensions) 进行高级环境自定义。

与扩展结尾日志和捆绑日志非常相似，可以使用配置文件来扩展日志轮换。只要 Elastic Beanstalk 轮换自己的日志并将其上传到 Amazon S3，它也轮换和上传您的附加日志。日志轮换扩展的行为因平台的操作系统而异。以下部分介绍两种情况。

在 Linux 上扩展日志轮换

如Linux 上的日志轮换设置中所述，Elastic Beanstalk 使用 `logrotate` 在 Linux 平台上轮换日志。如果配置应用程序的日志文件进行日志轮换，应用程序无需创建日志文件的副本。Elastic Beanstalk 将 `logrotate` 配置为对每个轮换创建应用程序日志文件的副本。因此，应用程序在未主动向日志文件写入内容时，必须使这些日志文件保持解锁状态。

在 Windows Server 上扩展日志轮换

在 Windows Server 中，如果配置应用程序的日志文件进行日志轮换，该应用程序必须定期轮这些日志文件。Elastic Beanstalk 查找其名称以配置的模式开头的文件，选取这些文件以便上传到 Amazon S3。此外，Amazon S3 忽略文件名中的句点，将句点之前的名称视为基本日志文件名。

Amazon S3 上传基本日志文件的所有版本（最新版本除外），因为它认为最新版本是活动的应用程序日志文件，可能处于锁定状态。因此，应用程序可使活动日志文件在轮换之间保持锁定状态。

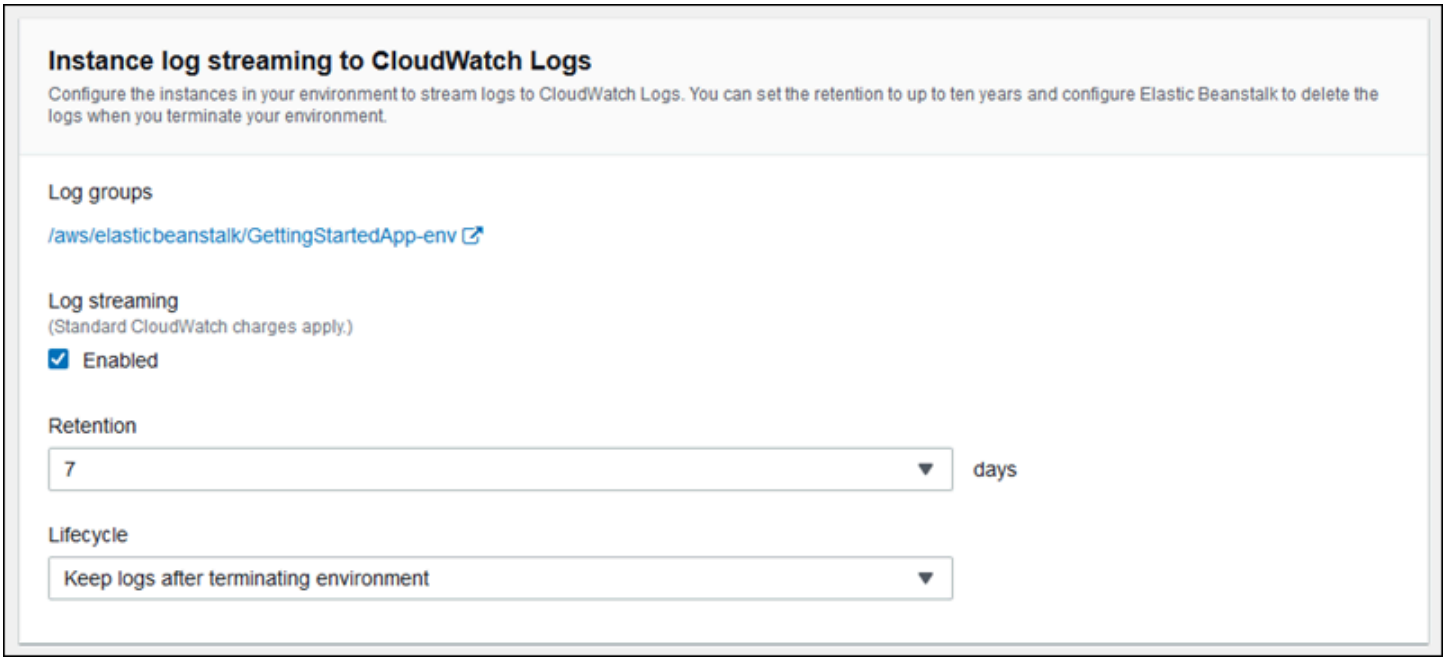
例如，应用程序写入名为 `my_log.log` 的日志文件，您在 `.conf` 文件中指定该名称。应用程序定期轮换该文件。在 Elastic Beanstalk 轮换周期中，它在日志文件的文件夹中找到以下文件：`my_log.log`、`my_log.0800.log`、`my_log.0830.log`。Elastic Beanstalk 将所有这些文件都视为基本名称 `my_log` 的版本。文件 `my_log.log` 的修改时间最晚，因此，Elastic Beanstalk 仅上传其他两个文件：`my_log.0800.log` 和 `my_log.0830.log`。

将日志文件流式传输到 Amazon CloudWatch Logs

您可以在 Elastic Beanstalk 控制台中或使用配置选项，将环境配置为流式传输日志到 Amazon CloudWatch Logs。借助 CloudWatch Logs，您环境中的每个实例都会将日志流式传输到日志组。对于日志组，您可以将其配置为保留数周或数年（即使在环境终止后）。

流式传输的日志集根据环境而有所不同，但始终包含 `eb-engine.log` 和来自在应用程序前运行的 nginx 或 Apache 代理服务器的访问日志。

您可以使用 Elastic Beanstalk 控制台，在创建环境期间或者为现有环境配置日志流。在以下示例中，日志最多保存七天，甚至在环境已终止时也是如此。



下面的配置文件启用了保留期为 180 天（即使环境已终止）的日志流。

Example .ebextensions/log-streaming.config

```
option_settings:
  aws:elasticbeanstalk:cloudwatch:logs:
    StreamLogs: true
    DeleteOnTerminate: false
    RetentionInDays: 180
```

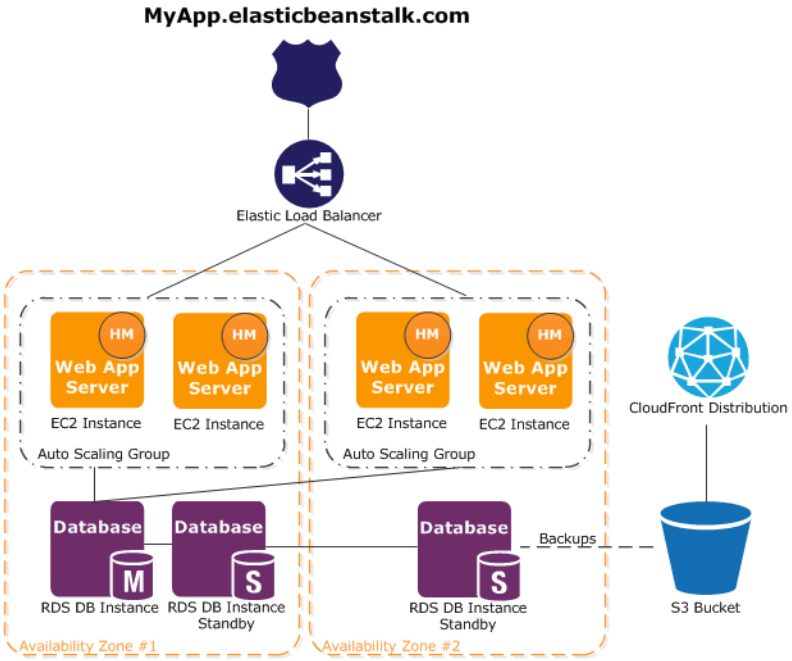
将 Elastic Beanstalk 与其他 AWS 服务结合使用

为了实施您的应用程序的环境，Elastic Beanstalk 将管理其他 AWS 服务的资源或使用其功能。此外，Elastic Beanstalk 与不直接作为您的环境的一部分使用的 AWS 服务集成。本节中的主题介绍了很多将这些附加服务与您的 Elastic Beanstalk 应用程序结合使用的方法。

主题

架构概述

下图演示了跨越多可用区的 Elastic Beanstalk 的示例架构，该架构与 Amazon CloudFront、Amazon Simple Storage Service (Amazon S3) 和 Amazon Relational Database Service (Amazon RDS) 等其他 AWS 产品配合使用。



要规划容错，最好具有 N+1 Amazon EC2 实例，然后将实例分布到多可用区域中。如果一个可用区域出现故障，您仍可使其他 Amazon EC2 实例在另一可用区域内运行。您可以调整 Amazon EC2 Auto Scaling，以允许使用最小数量的实例和多可用区。有关如何执行此操作的说明，请参阅 Elastic Beanstalk 环境的 Auto Scaling 组。有关构建容错应用程序的更多信息，请转到在 AWS 上构建容错的应用程序。

以下部分更为详细地描述了与 Amazon CloudFront、Amazon CloudWatch、Amazon DynamoDB、Amazon ElastiCache、Amazon RDS、Amazon Route 53、Amazon Simple Storage Service、Amazon VPC 和 IAM 的集成。

将 Elastic Beanstalk 和 Amazon CloudFront 结合使用

Amazon CloudFront 是一种将静态和动态 Web 内容（例如 .html、.css、.php、图像及媒体文件）加速分配给最终用户的 Web 服务。CloudFront 通过全球节点网络传送内容。当最终用户请求您用 CloudFront 提供的内容时，用户会被引导到延迟最低的节点，以便传送的内容具有最佳性能。如果内容已经在该节点中，CloudFront 将立即传送。如果内容目前不在该节点中，CloudFront 将从 Amazon S3 存储段或 HTTP 服务器（例如，web 服务器）对其进行检索，而且您已经确定该 Amazon S3 存储段或 HTTP 服务器为内容最终版本的源。

创建并部署 Elastic Beanstalk 应用程序之后，您可以注册 CloudFront，并开始使用 CloudFront 分发您的内容。有关 CloudFront 的更多信息，请参阅 Amazon CloudFront 开发人员指南。

使用 AWS CloudTrail 记录 Elastic Beanstalk API 调用

Elastic Beanstalk 与 AWS CloudTrail 集成，后者是一项提供用户、角色或AWS服务在 Elastic Beanstalk 中所采取操作的记录的服务。CloudTrail 将 Elastic Beanstalk 的所有 API 调用作为事件捕获，包括来自 Elastic Beanstalk 控制台、EB CLI 以及从代码到 Elastic Beanstalk API 的调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 Elastic Beanstalk 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history (事件历史记录) 中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 Elastic Beanstalk 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅 AWS CloudTrail 用户指南。

CloudTrail 中的 Elastic Beanstalk 信息

在您创建AWS账户时，将在该账户上启用 CloudTrail。当 Elastic Beanstalk 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他AWS服务事件一起保存在 Event history (事件历史记录) 中。您可以在AWS账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录AWS账户中的事件（包括 Elastic Beanstalk 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪记录来自 AWS 分区中的所有区域的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

CloudTrail 记录所有 Elastic Beanstalk 操作，AWS Elastic Beanstalk API 参考中介绍了这些操作。例如，对 `DescribeApplications`、`UpdateEnvironment` 和 `ListTagsForResource` 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员的信息。身份信息帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。

- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他AWS服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Elastic Beanstalk 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目演示了名为 `intern` 的 IAM 用户为 `sample-app` 应用程序中的 `sample-env` 环境调用的 `UpdateEnvironment` 操作。

```
{
  "Records": [{
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "AIXDAYQEXAMPLEUMLYNGL",
      "arn": "arn:aws:iam::123456789012:user/intern",
      "accountId": "123456789012",
      "accessKeyId": "ASXIAGXEXAMPLEQULKNXV",
      "userName": "intern",
      "sessionContext": {
        "attributes": {
          "mfaAuthenticated": "false",
          "creationDate": "2016-04-22T00:23:24Z"
        }
      },
      "invokedBy": "signin.amazonaws.com"
    },
    "eventTime": "2016-04-22T00:24:14Z",
    "eventSource": "elasticbeanstalk.amazonaws.com",
    "eventName": "UpdateEnvironment",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "255.255.255.54",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
      "applicationName": "sample-app",
```

```
      "environmentName": "sample-env",
      "optionSettings": []
    },
    "responseElements": null,
    "requestID": "84ae9ecf-0280-17ce-8612-705c7b132321",
    "eventID": "e48b6a08-c6be-4a22-99e1-c53139cbfb18",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }]
}
```


将 Elastic Beanstalk 和 Amazon CloudWatch 结合使用

Amazon CloudWatch 可让您监控、管理和发布各种指标，以及根据指标中的数据配置警报操作。Amazon CloudWatch 监控可让您收集、分析和查看系统和应用程序指标，从而能够更迅速地、更有信心地进行做出操作和业务决定。

您可以使用 Amazon CloudWatch 收集 Amazon Web Services（AWS）资源的相关指标 – 例如 Amazon EC2 实例的性能。您还可直接向 Amazon CloudWatch 发布自己的指标。Amazon CloudWatch 警报可帮助您更轻松地实施决策，因为可让您发送通知，或者按照定义的规则自动更改正在监控的资源。例如，您可创建代您启动 Amazon EC2 Auto Scaling 和 Amazon Simple Notification Service (Amazon SNS) 操作的警报。

Elastic Beanstalk 会自动使用 Amazon CloudWatch 帮助您监控应用程序和环境状态。您可导航到 Amazon CloudWatch 控制台，以查看您的控制面板，并对所有资源以及警报进行概览。您还可选择查看更多指标或添加自定义指标。

有关 Amazon CloudWatch 的更多信息，请转到 Amazon CloudWatch 开发人员指南。有关如何将 Amazon CloudWatch 与 Elastic Beanstalk 结合使用的示例，请参阅the section called “示例：使用自定义 Amazon CloudWatch 指标”。

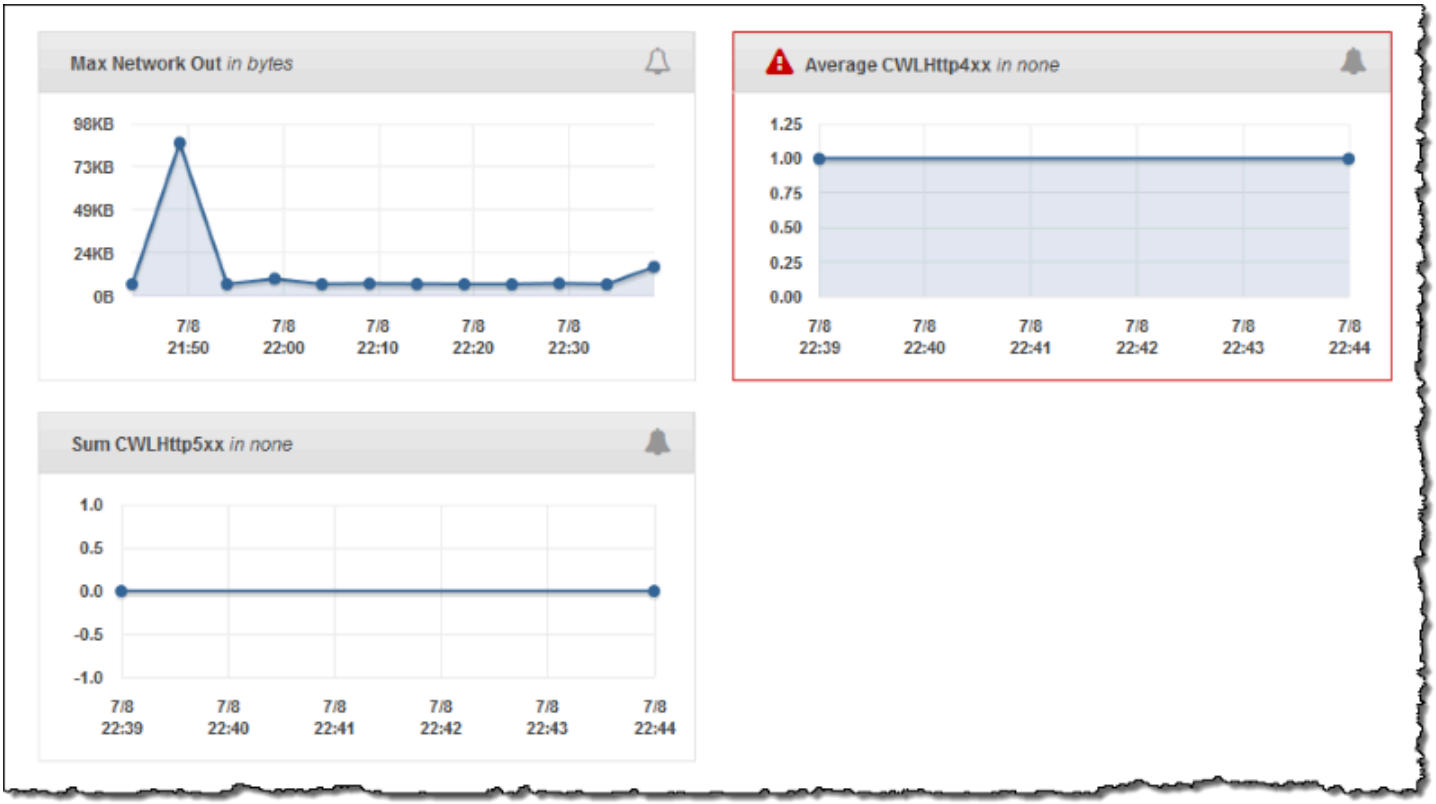
将 Elastic Beanstalk 和 Amazon CloudWatch Logs 配合使用

利用 CloudWatch Logs，您可以监控和存档您的环境的 Amazon EC2 实例中的 Elastic Beanstalk 应用程序、系统和自定义日志文件。您还可以配置警报，这些警报使您更容易根据指标筛选器提取的特定日志流事件做出响应。在环境中每个 Amazon EC2 实例上安装的 CloudWatch Logs 代理会针对您配置的每个日志组，将指标数据点发布到 CloudWatch 服务。每个日志组都应用自己的筛选模式，以确定要作为数据点发送到 CloudWatch 的日志流事件。属于同一日志组的各日志流具有相同的保留、监控和访

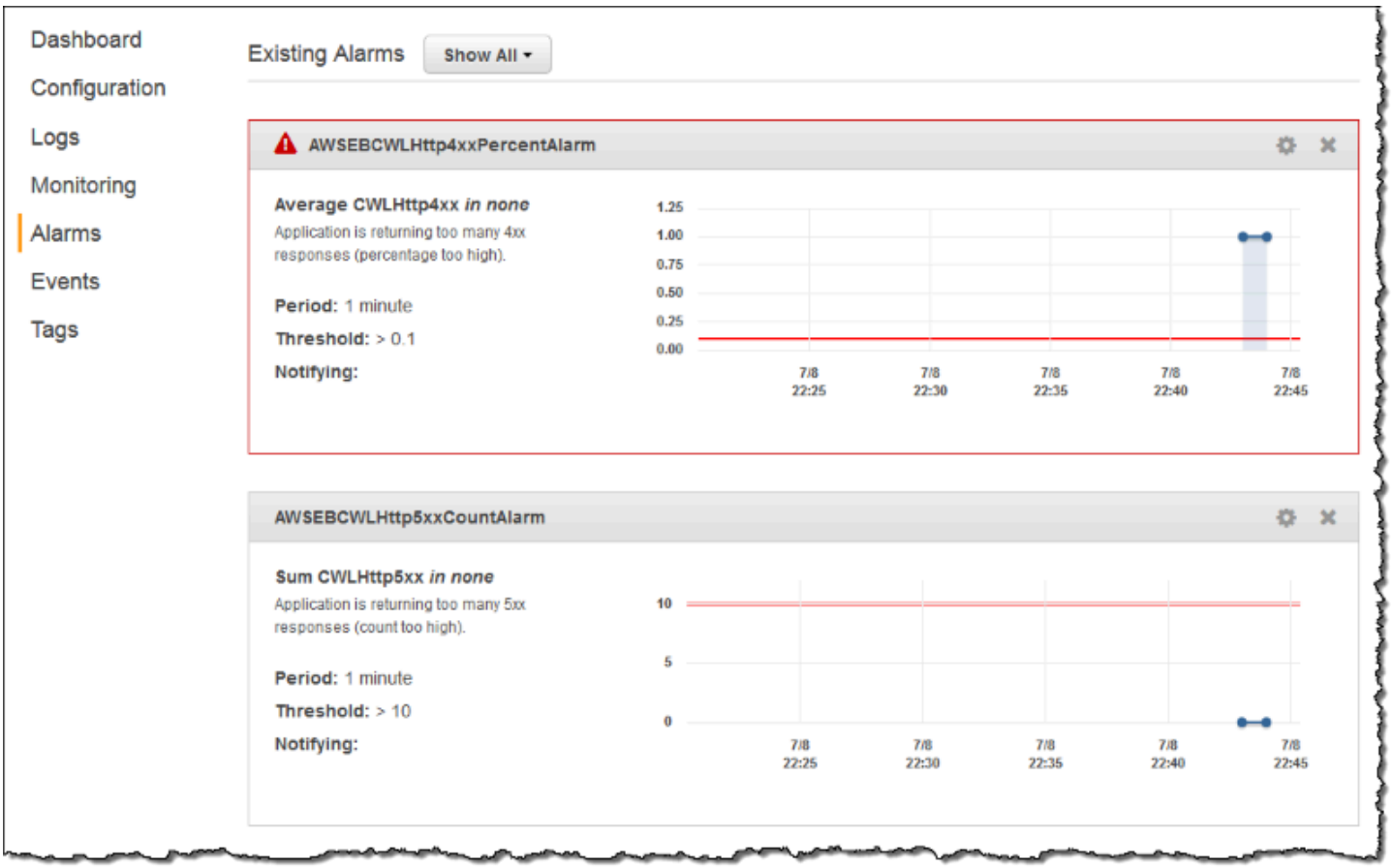
问控制设置。您可以配置 Elastic Beanstalk 自动将日志流式传输到 CloudWatch 服务，如将实例日志流式传输到 CloudWatch Logs 中所述。有关 CloudWatch Logs 的更多信息，包括术语和概念，请参阅 Amazon CloudWatch Logs 用户指南。

除了实例日志之外，如果您为环境启用增强型运行状况，则可以将环境配置为将运行状况信息流式传输至 CloudWatch Logs。请参阅将 Elastic Beanstalk 环境运行状况信息流式传输到 Amazon CloudWatch Logs。

下图显示了配置有 CloudWatch Logs 集成的环境的 Monitoring (监控) 页面和图表。此环境中的示例指标名为 CWLHttp4xx 和 CWLHttp5xx。其中一个图表显示了 CWLHttp4xx 指标根据配置文件指定的条件触发了警报。



下图显示了名为 AWSEBCWLHttp4xxPercentAlarm 和 AWSEBCWLHttp5xxCountAlarm 的示例警报（分别对应于 CWLHttp4xx 和 CWLHttp5xx 指标）的 Alarms（警报）页面和图表。



主题

- 实例日志流式传输到 CloudWatch Logs 的先决条件
- Elastic Beanstalk 如何设置 CloudWatch Logs
- 将实例日志流式传输到 CloudWatch Logs
- CloudWatch Logs 集成故障排除
- 将 Elastic Beanstalk 环境运行状况信息流式传输到 Amazon CloudWatch Logs

实例日志流式传输到 CloudWatch Logs 的先决条件

要启用日志从环境的 Amazon EC2 实例流式传输到 CloudWatch Logs，您必须满足以下条件。

- 平台 – 由于此功能仅在此版本推出时或之后发布的平台版本中可用，因此，如果您使用的是早期平台版本，请将环境更新为当前环境。
- 如果您的 Elastic Beanstalk 实例配置文件中没有 AWSElasticBeanstalkWebTier 或 AWSElasticBeanstalkWorkerTier Elastic Beanstalk 托管策略，则您必须向配置文件中添加以下内容来启用此功能。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:PutLogEvents",
      "logs:CreateLogStream"
    ],
    "Resource": [
    "*"
    ]
  }
  ]
}
```

Elastic Beanstalk 如何设置 CloudWatch Logs

Elastic Beanstalk 会在其创建的每个实例上安装一个具有默认配置设置的 CloudWatch 日志代理。在 CloudWatch Logs 代理参考中了解更多信息。

当您启动实例日志流式传输到 CloudWatch Logs 时，Elastic Beanstalk 会将日志文件从您环境的实例发送至 CloudWatch Logs。不同的平台将流式传输不同的日志。下表按平台列出了日志。

平台/平台分支	日志
Docker / 平台分支：在 64 位 Amazon Linux 2 上运行的 Docker	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/docker • /var/log/docker-events.log • /var/log/eb-docker/containers/eb-current-app/stdouterr.log • /var/log/nginx/access.log • /var/log/nginx/error.log
Docker /	• /var/log/docker-events.log • /var/log/eb-ecs-mgr.log • /var/log/eb-engine.log

平台/平台分支	日志
平台分支：在 64 位 Amazon Linux 2 上运行的 ECS	• /var/log/eb-hooks.log • /var/log/ecs/ecs-agent.log • /var/log/ecs/ecs-init.log
Go .NET Core on Linux Java /平台分支：在 64 位 Amazon Linux 2 上运行的 Corretto	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/web.stdout.log • /var/log/nginx/access.log • /var/log/nginx/error.log
Node.js Python	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/web.stdout.log • /var/log/httpd/access_log • /var/log/httpd/error_log • /var/log/nginx/access.log • /var/log/nginx/error.log
Tomcat PHP	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/httpd/access_log • /var/log/httpd/error_log • /var/log/nginx/access.log • /var/log/nginx/error.log
Windows Server 上的 .NET	• C:\inetpub\logs\LogFiles\W3SVC1\u_ex*.log • C:\Program Files\Amazon\ElasticBeanstalk\logs\AWSDeployment.log • C:\Program Files\Amazon\ElasticBeanstalk\logs\Hooks.log

平台/平台分支	日志
Ruby	• /var/log/eb-engine.log • /var/log/eb-hooks.log • /var/log/puma/puma.log • /var/log/web.stdout.log • /var/log/nginx/access.log • /var/log/nginx/error.log

Amazon Linux AMI 平台上的日志文件

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。

下表按平台列出从平台分支上基于 Amazon Linux AMI（Amazon Linux 2 以前的版本）的实例流式传输的日志文件。

平台/平台分支	日志
Docker / 平台分支：在 64 位 Amazon Linux 上运行的 Docker	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/docker-events.log • /var/log/docker • /var/log/nginx/access.log • /var/log/eb-docker/containers/eb-current-app/stdouterr.log
Docker / 平台分支：在 64 位 Amazon Linux 上运行的多容器 Docker	• /var/log/eb-activity.log • /var/log/ecs/ecs-init.log • /var/log/eb-ecs-mgr.log

平台/平台分支	日志
	• /var/log/ecs/ecs-agent.log • /var/log/docker-events.log
Glassfish（预配置的 Docker）	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/docker-events.log • /var/log/docker • /var/log/nginx/access.log
Go	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/nginx/access.log
Java / 平台分支：在 64 位 Amazon Linux 上运行的 Java 8 平台分支：在 64 位 Amazon Linux 上运行的 Java 7	• /var/log/eb-activity.log • /var/log/nginx/access.log • /var/log/nginx/error.log • /var/log/web-1.error.log • /var/log/web-1.log
Tomcat	• /var/log/eb-activity.log • /var/log/httpd/error_log • /var/log/httpd/access_log • /var/log/nginx/error_log • /var/log/nginx/access_log
Node.js	• /var/log/eb-activity.log • /var/log/nodejs/nodejs.log • /var/log/nginx/error.log • /var/log/nginx/access.log • /var/log/httpd/error.log • /var/log/httpd/access.log

平台/平台分支	日志
PHP	• /var/log/eb-activity.log • /var/log/httpd/error_log • /var/log/httpd/access_log
Python	• /var/log/eb-activity.log • /var/log/httpd/error_log • /var/log/httpd/access_log • /opt/python/log/supervisord.log
Ruby / 平台分支：基于在 64 位 Amazon Linux 上运行的 Ruby 的 Puma	• /var/log/eb-activity.log • /var/log/nginx/error.log • /var/log/puma/puma.log • /var/log/nginx/access.log
Ruby / 平台分支：基于在 64 位 Amazon Linux 上运行的 Ruby 的 Passenger	• /var/log/eb-activity.log • /var/app/support/logs/passenger.log • /var/app/support/logs/access.log • /var/app/support/logs/error.log

Elastic Beanstalk 针对其流式传输的各种日志文件在 CloudWatch Logs 中配置日志组。要从 CloudWatch Logs 中检索特定日志文件，您必须知道相应日志组的名称。日志组命名方案取决于平台的操作系统。

对于 Linux 平台，为实例上日志文件位置加上前缀 `/aws/elasticbeanstalk/`*`environment_name`* 以获得日志组名称。例如，要检索文件 `/var/log/nginx/error.log`，请指定日志组 `/aws/elasticbeanstalk/`*`environment_name`*`/var/log/nginx/error.log`。

有关 Windows 平台，请参阅下表以了解每个日志文件对应的日志组。

实例上日志文件	日志组
C:\Program Files\Amazon\ElasticBeanstalk\logs\AWSDeployment.log	/aws/elasticbeanstalk/<environment-name>/EBDeploy-Log
C:\Program Files\Amazon\ElasticBeanstalk\logs\Hooks.log	/aws/elasticbeanstalk/<environment-name>/EBHooks-Log
C:\inetpub\logs\LogFiles (整个目录)	/aws/elasticbeanstalk/<environment-name>/IIS-Log

将实例日志流式传输到 CloudWatch Logs

您可以使用 Elastic Beanstalk 控制台、EB CLI 或配置选项，启用实例日志流式传输到 CloudWatch Logs。

在启用它之前，请设置 IAM 权限以结合使用 CloudWatch Logs 代理。您可以将以下自定义策略挂载到分配给您的环境的实例配置文件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

使用 Elastic Beanstalk 控制台进行实例日志流式传输

将实例日志流式传输到 CloudWatch Logs

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在 Instance log streaming to CloudWatch Logs (实例日志流式传输到 CloudWatch Logs) 下：
   - 启用 Log streaming (日志流式传输)。
   - 将 Retention (保留) 设置为保存日志的天数。
   - 选择 Lifecycle (生命周期) 设置，该设置确定日志是否在环境终止时进行保存。
6. 要保存更改，请选择页面底部的 Apply（应用）。

在启用日志流式传输后，您可以返回到软件配置类别或页面并查找日志组链接。单击此链接可在 CloudWatch 控制台中查看您的日志。

使用 EB CLI 进行实例日志流式传输

要使用 EB CLI 启用实例日志流式传输到 CloudWatch Logs，请使用 eb logs 命令。

```
$ eb logs --cloudwatch-logs enable
```

您也可以使用 eb logs 来检索 CloudWatch Logs 中的日志。您可以检索环境的所有实例日志，也可以使用该命令的多个选项来指定要检索的日志的子集。例如，以下命令检索您的环境中一组完整的实例日志，并且将它们保存到 `.elasticbeanstalk/logs` 下的目录。

```
$ eb logs --all
```

具体而言，`--log-group` 选项可让您检索特定日志组的实例日志，对应于特定实例上日志文件。为此，您需要了解对应于要检索的日志文件的日志组的名称。您可以在 Elastic Beanstalk 如何设置 CloudWatch Logs 中找到此信息。

使用配置文件进行日志流式传输

在创建或更新环境时，您可以使用配置文件来设置和配置实例日志流式传输到 CloudWatch Logs。以下示例配置文件启用默认实例日志流式传输。Elastic Beanstalk 为您的环境平台流式传输一组默认日志文件。要使用此示例，请将该文本复制到应用程序源代码包的顶层 `.ebextensions` 目录中扩展名为 `.config` 的文件。

```
option_settings:
  - namespace: aws:elasticbeanstalk:cloudwatch:logs
    option_name: StreamLogs
    value: true
```

自定义日志文件流式传输

Elastic Beanstalk 与 CloudWatch Logs 集成不直接支持对您的应用程序生成的自定义日志文件进行流式传输。要流式传输自定义日志，请使用配置文件直接安装 CloudWatch Logs 代理和配置要推送的文件。有关示例配置文件，请参阅 `logs-streamtocloudwatch-linux.config`。

> **Note**
>
> 该示例在 Windows 平台上不起作用。

有关配置 CloudWatch Logs 的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的 CloudWatch Logs 代理参考。

CloudWatch Logs 集成故障排除

如果您无法在 CloudWatch Logs 中找到所期望的环境的一些实例日志，则可以调查以下常见问题：

- 您的 IAM 角色缺少所需 IAM 权限。
- 您在不支持 CloudWatch Logs 的 AWS 区域 中启动了您的环境。
- 您的其中一个自定义日志文件不存在于您指定的路径中。

将 Elastic Beanstalk 环境运行状况信息流式传输到 Amazon CloudWatch Logs

如果您为环境启用增强型运行状况报告，则可以将环境配置为将运行状况信息流式传输至 CloudWatch Logs。此流式传输独立于 Amazon EC2 实例日志流式传输。本主题介绍环境运行状况信息流式传输。有关实例日志流式传输的信息，请参阅将 Elastic Beanstalk 和 Amazon CloudWatch Logs 配合使用。

当您配置环境运行状况流式传输时，Elastic Beanstalk 会为环境运行状况创建一个 CloudWatch Logs 日志组。日志组的名称为 `/aws/elasticbeanstalk/`*`environment-name`*`/environment-health.log`。在此日志组内，Elastic Beanstalk 创建名为 *`YYYY-MM-DD`*`#`*`<hash-suffix>`* 的日志流（每个日期可能存在多个日志流）。

当环境的运行状况发生更改时，Elastic Beanstalk 会将记录添加到运行状况日志流。该记录表示运行状况转换 - 新状态以及发生更改的原因的描述。例如，环境的状态可能由于负载均衡器发生故障而更改为“严重”。有关增强型运行状况的描述，请参阅运行状况颜色和状态。

环境运行状况流式传输到 CloudWatch Logs 的先决条件

要启用环境运行状况流式传输到 CloudWatch Logs，您必须满足以下条件：

- 平台 – 您必须使用支持增强型运行状况报告的平台版本。
- 权限 – 您必须向 Elastic Beanstalk 授予与日志记录相关的特定权限，以便它可以代表您执行操作以流式传输您的环境的运行状况信息。如果您的环境未使用 Elastic Beanstalk 为其创建的服务角色 `aws-elasticbeanstalk-service-role` 或您的账户的服务相关角色 `AWSServiceRoleForElasticBeanstalk`，请务必向您的自定义服务角色添加以下权限。

```
{
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk/*:log-stream:*"
}
```

将环境运行状况日志流式传输到 CloudWatch Logs

您可以使用 Elastic Beanstalk 控制台、EB CLI 或配置选项，启用将环境运行状况流式传输到 CloudWatch Logs。

使用 Elastic Beanstalk 控制台进行环境运行状况日志流式传输

将环境运行状况日志流式传输到 CloudWatch Logs

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Monitoring (监控) 配置类别中，选择 Edit (编辑)。
5. 在 Health reporting (运行状况报告) 下，请确保报告 System (系统) 设置为 Enhanced (增强型)。
6. 在 Health event streaming to CloudWatch Logs (运行状况事件流式传输到 CloudWatch Logs) 下

   - 启用 Log streaming (日志流式传输)。
   - 将 Retention (保留) 设置为保存日志的天数。
   - 选择 Lifecycle (生命周期) 设置，该设置确定日志是否在环境终止时进行保存。
7. 要保存更改，请选择页面底部的 Apply（应用）。

在启用日志流式传输后，您可以返回到监控配置类别或页面并查找日志组链接。单击此链接可在 CloudWatch 控制台中查看您的环境运行状况日志。

使用 EB CLI 进行环境运行状况日志流式传输

要使用 EB CLI 启用环境运行状况日志流式传输到 CloudWatch Logs，请使用 eb logs 命令。

```
$ eb logs --cloudwatch-logs enable --cloudwatch-log-source environment-health
```

您也可以使用 eb logs 来检索 CloudWatch Logs 中的日志。例如，以下命令检索您的环境中所有的运行状况日志，并且将它们保存到 `.elasticbeanstalk/logs` 下的目录。

```
$ eb logs --all --cloudwatch-log-source environment-health
```

使用配置文件进行环境运行状况日志流式传输

在创建或更新环境时，您可以使用配置文件来设置和配置实例环境运行状况日志流式传输到 CloudWatch Logs。要使用下列示例，请将该文本复制到应用程序源包的顶层 `.config` 目录中扩展名为 `.ebextensions` 的文件。该示例将 Elastic Beanstalk 配置为启用环境运行状况日志流式传输，在终止环境后保留日志，并且将日志保存 30 天。

Example 运行状况流式传输配置文件

```
#############################################################################
##  Sets up Elastic Beanstalk to stream environment health information
##  to Amazon CloudWatch Logs.
##  Works only for environments that have enhanced health reporting enabled.
#############################################################################

option_settings:
  aws:elasticbeanstalk:cloudwatch:logs:health:
    HealthStreamingEnabled: true
    ### Settings below this line are optional.
    # DeleteOnTerminate: Delete the log group when the environment is
    # terminated. Default is false. If false, the health data is kept
    # RetentionInDays days.
    DeleteOnTerminate: false
    # RetentionInDays: The number of days to keep the archived health data
    # before it expires, if DeleteOnTerminate isn't set. Default is 7 days.
    RetentionInDays: 30
```

有关选项默认值和有效值，请参阅 aws:elasticbeanstalk:cloudwatch:logs:health。

将 Elastic Beanstalk 与 Amazon EventBridge 结合使用

使用 Amazon EventBridge，您可以设置事件驱动型规则，以监控您的 Elastic Beanstalk 资源并启动使用其他AWS服务的目标操作。例如，每当生产环境的运行状况变为 Warning（警告）状态时，您可以通过向 Amazon SNS 主题发出信号来设置发送电子邮件通知的规则。或者，您可以将 Lambda 函数设置为每当环境的运行状况变为 Degraded（降级）或 Severe（严重）状态时，将通知传递给 Slack。

您可以在 Amazon EventBridge 中创建规则，以处理以下任何 Elastic Beanstalk 事件：

- 环境操作的状态更改（包括创建、更新和终止操作）。该事件指定状态更改是已启动、成功还是失败。
- 其他资源的状态更改。除了环境之外，系统监控的其他资源包括负载均衡器、Auto Scaling 组和实例。
- 环境的运行状况转变。该事件表明环境运行状况从一种运行状况过渡到另一种状况。
- 托管更新的状态更改。该事件指定状态更改是已启动、成功还是失败。

要捕获您感兴趣的特定 Elastic Beanstalk 事件，请定义 EventBridge 可用于检测事件的特定事件模式。事件模式与它们匹配的事件具有相同的结构。模式引用了您要匹配的字段，并提供您所查找的值。尽最大努力发出事件。在正常运行环境下，它们被近乎实时地从 Elastic Beanstalk 发送到 EventBridge。但是，可能会出现延迟或阻止事件交付的情况。

有关 Elastic Beanstalk 事件中包含的字段列表及其可能的字符串值，请参阅Elastic Beanstalk 事件字段映射。有关 EventBridge 规则如何处理事件模式的信息，请参阅 EventBridge 中的事件和事件模式。

使用 EventBridge 监控 Elastic Beanstalk 资源

使用 EventBridge，您可以创建规则来定义 Elastic Beanstalk 为其资源发出事件时要采取的操作。例如，您可以创建一个规则，只要环境状态发生变化就向您发送电子邮件。

EventBridge 控制台具有用于构建 Elastic Beanstalk 事件模式的预定义模式选项。如果您在创建规则时在 EventBridge 控制台中选择此选项，则可以快速构建 Elastic Beanstalk 事件模式。您只需选择事件字段和值即可。在您进行选择时，控制台将构建并显示事件模式。或者，您可以手动编辑构建的事件模式，并将其另存为自定义模式。控制台还为您提供显示详细的 Sample Event（示例事件）选项，您可以将其复制并粘贴到您正在构建的事件模式中。

如果您希望键入或复制事件模式并将其粘贴到 EventBridge 控制台中，则可以选择在控制台中使用 Custom pattern（自定义模式）选项。通过这样做，您无需完成前面描述的选择字段和值的步骤。本主题提供了可以使用的事件匹配模式和 Elastic Beanstalk 事件的示例。

要为资源事件创建规则

1. 通过具有 EventBridge 和 Elastic Beanstalk 使用权限的账户登录AWS。
2. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
3. 在导航窗格中，选择 Rules (规则)。
4. 选择 Create rule (创建规则)。
5. 输入规则的 Name (名称) 和“Description (描述)”（可选）。

6. 对于 Event bus（事件总线），选择 default（默认）。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
7. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
8. 选择 Next (下一步)。
9. 对于 Event source（事件源），选择 AWS 事件或 EventBridge 合作伙伴事件。
10. （可选）对于 Sample event（示例事件），选择 AWS events（AWS 事件）。在搜索字段中输入 Elastic Beanstalk。这将提供一个示例 Elastic Beanstalk 事件列表，您可以从中选择显示的事件。此步骤仅显示您可以参考的示例事件。它不影响规则创建的结果。本主题后面的 Elastic Beanstalk 事件示例 部分提供了相同类型事件的示例。
11. 在 Event pattern（事件模式）部分，选择 Event pattern form（事件模式表单）。

> **Note**
>
> 如果您已经有事件模式的文本，并且不需要 EventBridge 控制台为您构建它，请选择 Custom pattern (JSON editor)（自定义模式（JSON 编辑器））。然后，您可以手动输入或将文本复制粘贴到 Event pattern（事件模式）框中。选择 Next（下一步），然后转到关于进入目标的步骤。

12. 对于 Event source（事件源），选择 AWS services（AWS 服务）。
13. 对于 AWS service（AWS 服务），选择 Elastic Beanstalk。
14. 对于 Event type（事件类型），选择 Status Change（状态更改）。
15. 本步骤介绍了如何使用 Elastic Beanstalk 的 detail type（详细信息类型）、status（状态）和 severity（严重性）事件字段。当您选择这些字段和要匹配的值时，控制台将构建并显示事件模式。

    - 如果您为 Specific detail type(s)（特定详细信息类型）仅选择一个值，则可以为层次结构中的下一个字段选择一个或多个值。
    - 如果您为 Specific detail type(s)（特定详细信息类型）选择多个值，则可以为层次结构中的下一个字段选择特定值。这样可防止事件模式中各字段之间的不明确匹配逻辑。

    environment（环境）事件字段不受此层次结构的影响，因此它将按下一步中的说明显示。
16. 对于环境，选择 Any environment（任意环境）或 Specific environment(s)（特定环境）。

- 如果您选择 Specific environment(s)（特定环境），则可以从下拉列表中选择一个或多个环境。EventBridge 会将您在 EnvironmentName[ ] 列表中选择的所有环境添加到事件模式的 detail（详细信息）部分。然后，您的规则将筛选所有事件，以仅包括您选择的特定环境。
- 如果您选择 Any environment（任意环境），则不会向事件模式中添加任何环境。因此，您的规则不会根据环境筛选任何 Elastic Beanstalk 事件。

17. 选择 Next (下一步)。
18. 对于 Target types（目标类型），选择 AWS service（AWS 服务）。
19. 对于 Select a target（选择目标），选择从 Elastic Beanstalk 收到资源状态更改事件时要执行的目标操作。

    例如，您可以使用 Amazon Simple Notification Service (SNS) 主题在事件发生时发送电子邮件或短信。为此，您需要使用 Amazon SNS 控制台创建 Amazon SNS 主题。要了解更多信息，请参阅使用 Amazon SNS 发送用户通知。

    > **Important**
    >
    > 一些目标操作可能需要使用其他服务并会产生额外费用，例如 Amazon SNS 或 Lambda 服务。有关AWS定价的更多信息，请参阅https://aws.amazon.com/pricing/。某些服务是AWS免费使用套餐的一部分。如果您是新客户，则可免费试用这些服务。参阅 https://aws.amazon.com/free/ 了解更多信息。

20. （可选）选择 Add another target（添加另一个目标）来为事件规则指定其他目标操作。
21. 选择 Next (下一步)。
22. （可选）为规则输入一个或多个标签。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Amazon EventBridge 标签。
23. 选择 Next (下一步)。
24. 查看规则详细信息并选择 Create rule（创建规则）。

Elastic Beanstalk 事件模式示例

事件模式与它们匹配的事件具有相同的结构。模式引用了您要匹配的字段，并提供您所查找的值。

- 所有环境的运行状况更改

```
{
```

```
   "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Health status change"
    ]
}
```

- 以下环境的运行状况更改：myEnvironment1 和 myEnvironment2。此事件模式可针对这两个特定环境进行筛选，而之前未筛选的运行状态更改示例会为所有环境发送事件。

```
{"source": [
    "aws.elasticbeanstalk"
    ],
    "detail-type": [
        "Health status change"
    ],
    "detail": {
        "EnvironmentName": [
            "myEnvironment1",
            "myEnvironment2"
        ]
    }
}
```

- 针对所有环境的Elastic Beanstalk 资源状态更改

```
{
  "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Elastic Beanstalk resource status change"
    ]
}
```

- 对于以下环境，Elastic Beanstalk 资源状态更改为 Status Environment update failed（环境更新失败）和 Severity ERROR（错误）：myEnvironment1 和 myEnvironment2

```
{"source": [
    "aws.elasticbeanstalk"
    ],
    "detail-type": [
```

```
        "Elastic Beanstalk resource status change"
    ],
    "detail": {
        "Status": [
            "Environment update failed"
            ],
        "Severity": [
            "ERROR"
            ],
        "EnvironmentName": [
            "myEnvironment1",
            "myEnvironment2"
        ]
    }
}
```

- 负载均衡器、Auto Scaling 组和实例的其他资源状态更改

```
{
   "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Other resource status change"
    ]
}
```

- 所有环境的托管更新状态更改

```
{
   "source": [
    "aws.elasticbeanstalk"
  ],
  "detail-type": [
    "Managed update status change"
    ]
}
```

- 捕获 Elastic Beanstalk 的所有事件（不包括 `detail-type` 部分）

```
{
  "source": [
    "aws.elasticbeanstalk"
```

```
  ]
}
```

Elastic Beanstalk 事件示例

以下是资源状态更改 的 Elastic Beanstalk 事件示例：

```
{
   "version":"0",
   "id":"1234a678-1b23-c123-12fd3f456e78",
   "detail-type":"Elastic Beanstalk resource status change",
   "source":"aws.elasticbeanstalk",
   "account":"111122223333",
   "time":"2020-11-03T00:31:54Z",
   "region":"us-east-1",
   "resources":[
      "arn:was:elasticbeanstalk:us-east-1:111122223333:environment/myApplication/
myEnvironment"
   ],
   "detail":{
      "Status":"Environment creation started",
      "EventDate":1604363513951,
      "ApplicationName":"myApplication",
      "Message":"createEnvironment is starting.",
      "EnvironmentName":"myEnvironment",
      "Severity":"INFO"
   }
}
```

以下是运行状况更改 的 Elastic Beanstalk 事件示例：

```
{
   "version":"0",
   "id":"1234a678-1b23-c123-12fd3f456e78",
   "detail-type":"Health status change",
   "source":"aws.elasticbeanstalk",
   "account":"111122223333",
   "time":"2020-11-03T00:34:48Z",
   "region":"us-east-1",
   "resources":[
      "arn:was:elasticbeanstalk:us-east-1:111122223333:environment/myApplication/
myEnvironment"
```

```
   ],
   "detail":{
      "Status":"Environment health changed",
      "EventDate":1604363687870,
      "ApplicationName":"myApplication",
      "Message":"Environment health has transitioned from Pending to Ok. Initialization
 completed 1 second ago and took 2 minutes.",
      "EnvironmentName":"myEnvironment",
      "Severity":"INFO"
   }
}
```

Elastic Beanstalk 事件字段映射

下表将 Elastic Beanstalk 事件字段及其可能的字符串值映射到 EventBridge `detail-type` 字段。有关 EventBridge 如何处理服务的事件模式的更多信息，请参阅 EventBridge 中的事件和事件模式。

EventBridge 字段 detail-type	Elastic Beanstalk 字段 Status	Elastic Beanstalk 字段 Severity	Elastic Beanstalk 字段 Message
Elastic Beanstalk 资源状态更改	环境创建已启动	INFO	createEnvironment 正在启动。
	环境创建成功	INFO	createEnvironment 已成功完成。
	环境创建成功	INFO	启动的环境：<Environment Name>。但是，启动过程中存在问题。有关详细信息，请参阅事件日志。
	环境创建失败	ERROR (错误)	无法启动环境。
	环境更新已启动	INFO	环境更新正在启动。
	环境更新成功	INFO	环境更新已成功完成。

EventBridge 字段 detail-type	Elastic Beanstalk 字段 Status	Elastic Beanstalk 字段 Severity	Elastic Beanstalk 字段 Message
	环境更新失败	ERROR (错误)	无法部署配置。
	环境终止已开始	INFO	terminateEnvironment 正在启动。
	环境终止成功	INFO	terminateEnvironment 已成功完成。
	环境终止失败	INFO	由于至少有一个环境终止工作流失败，环境终止步骤失败。
其他资源状态更改	Auto Scaling 组已创建	INFO	createEnvironment 正在启动。
	Auto Scaling 组已删除	INFO	createEnvironment 正在启动。
	实例已添加	INFO	已将实例 [i-123456789a12b1234] 添加到环境。
	实例已删除	INFO	已从环境中删除实例 [i-123456789a12b1234]。
	负载均衡器已创建	INFO	已创建负载均衡器名称：<LB Name>
	负载均衡器已删除	INFO	删除以下名称的负载均衡器：<LB Name>
运行状况更改	环境运行状况已更改	信息/警告	环境运行状况已转为 <healthStatus>。

EventBridge 字段 detail-type	Elastic Beanstalk 字段 Status	Elastic Beanstalk 字段 Severity	Elastic Beanstalk 字段 Message
	环境运行状况已更改	信息/警告	环境运行状况已从 <healthStatus> 转为 <healthStatus>。
托管更新状态更改	托管更新已启动	INFO	托管平台更新正在进行中。
	托管更新失败	INFO	托管更新失败，在 %s 分钟后重试。

使用 查找和跟踪 Elastic Beanstalk 资源AWS Config

AWS Config 可以提供关于您的 AWS 账户中的 AWS 资源配置的详细信息。您可以查看资源的关联方式、获取配置更改的历史记录并了解关系和配置如何随时间的推移而变化。您可以使用 AWS Config 来定义规则，用于评估资源配置的数据合规性。

有多种 Elastic Beanstalk 资源类型与 集成AWS Config

- 应用程序
- 应用程序版本
- 环境

以下章节说明如何将 AWS Config 配置为记录这些类型的资源。

有关 AWS Config 的更多信息，请参阅 AWS Config 开发人员指南。有关定价信息，请参阅 AWS Config 定价信息页。

设置 AWS Config

要初次设置 AWS Config，请参阅 AWS Config 开发人员指南中的以下主题。

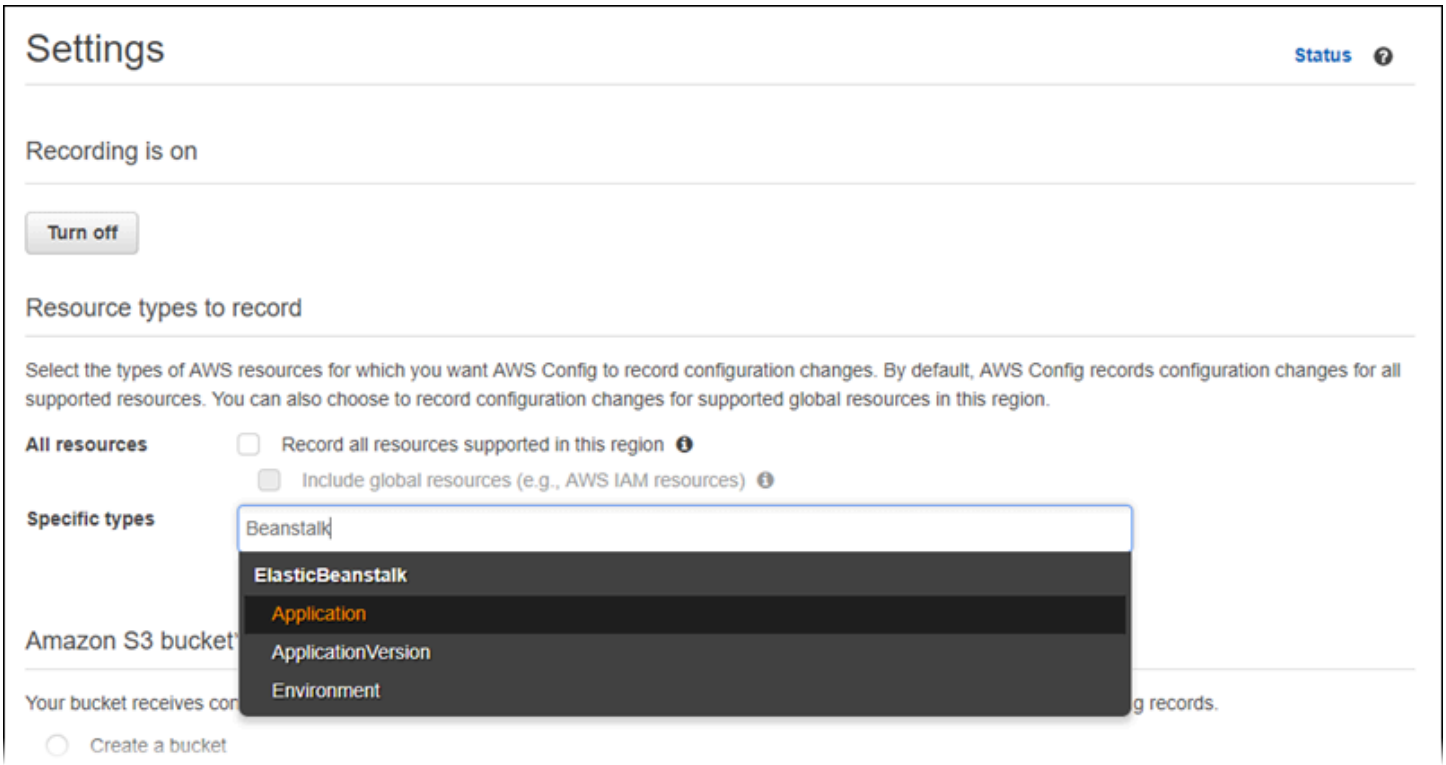
- 使用控制台设置 AWS Config
- 通过以下方式设置 AWS Config：AWS CLI

配置 AWS Config 以记录 Elastic Beanstalk 资源

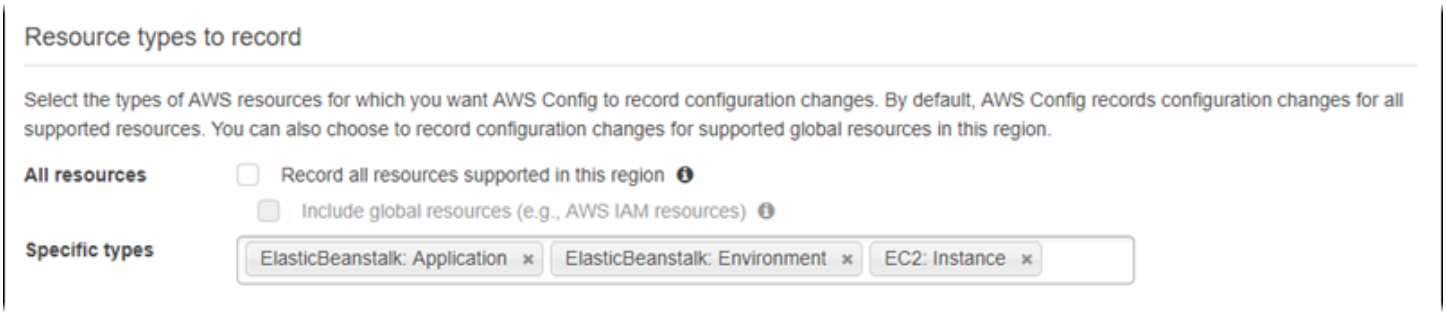
默认情况下，AWS Config 会记录在环境的运行区域中发现的所有受支持类型的区域性资源 的配置更改。您可以自定义 AWS Config，使其仅记录特定资源类型的更改或仅记录对全局资源 的更改。

例如，您可以将 AWS Config 配置为记录 Elastic Beanstalk 资源的更改以及 Elastic Beanstalk 为您启动的一部分其他 AWS 资源的更改。使用 AWS Config 控制台，您可以在 AWS Config Settings (设置) 页面的 Specific Types (特定类型) 字段中选择 Elastic Beanstalk 作为资源。在该位置，您可以选择记录任何 Elastic Beanstalk 资源类型：Application (应用程序)、ApplicationVersion 和 Environment (环境)。

下图显示了 AWS Config Settings（设置）页面，其中包含可以选择记录的 Elastic Beanstalk 资源类型：Application（应用程序）、ApplicationVersion 和 Environment（环境）。



在选择几个资源类型后，下面是 Specific types (特定类型) 列表的样子。



要了解区域 与全局 资源以及完整的自定义过程，请参阅选择 AWS Config 记录的资源。

在 AWS Config 控制台中查看 Elastic Beanstalk 配置详细信息

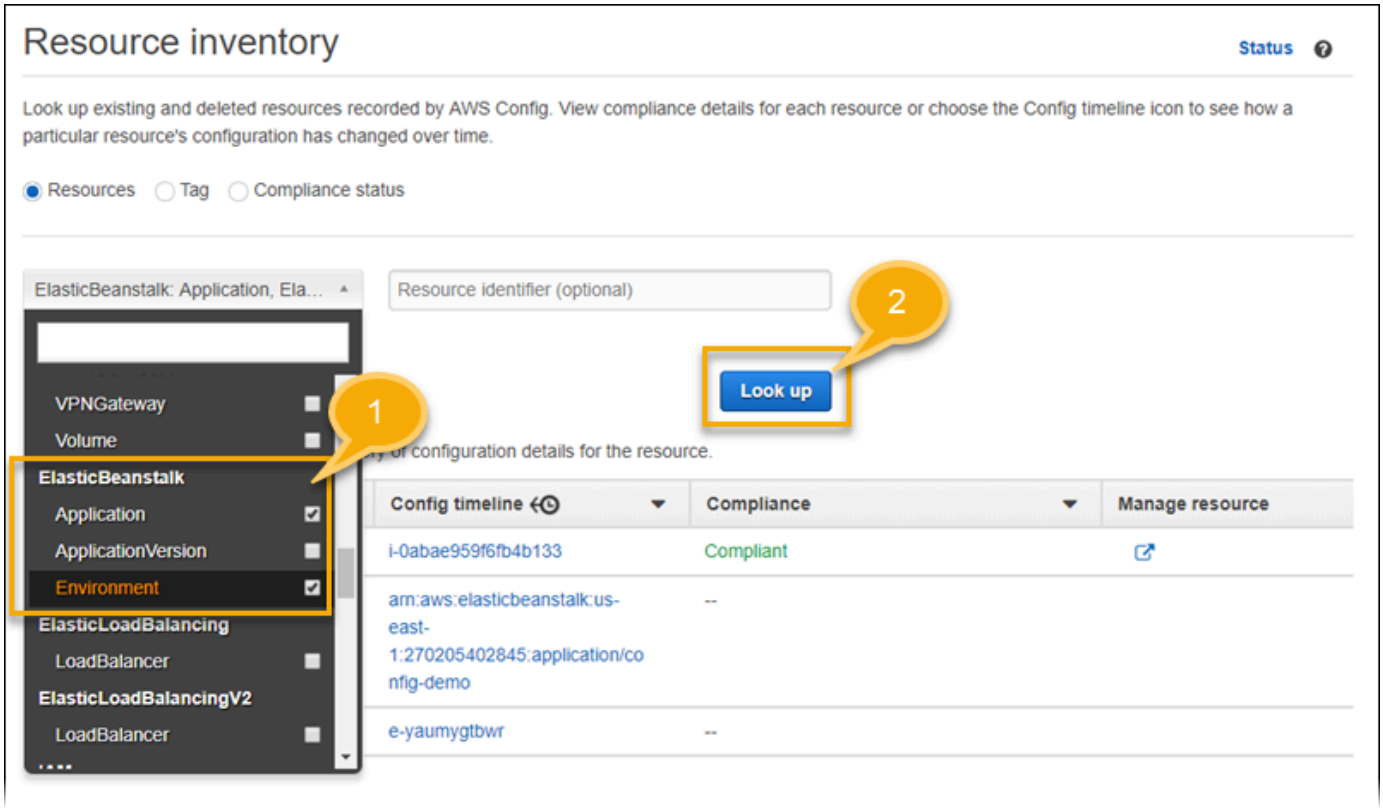
您可使用 AWS Config 控制台来查找 Elastic Beanstalk 资源，并获取有关其配置的当前和历史详细信息。以下示例说明如何查找有关 Elastic Beanstalk 环境的信息。

在 AWS Config 控制台中查找 Elastic Beanstalk 环境

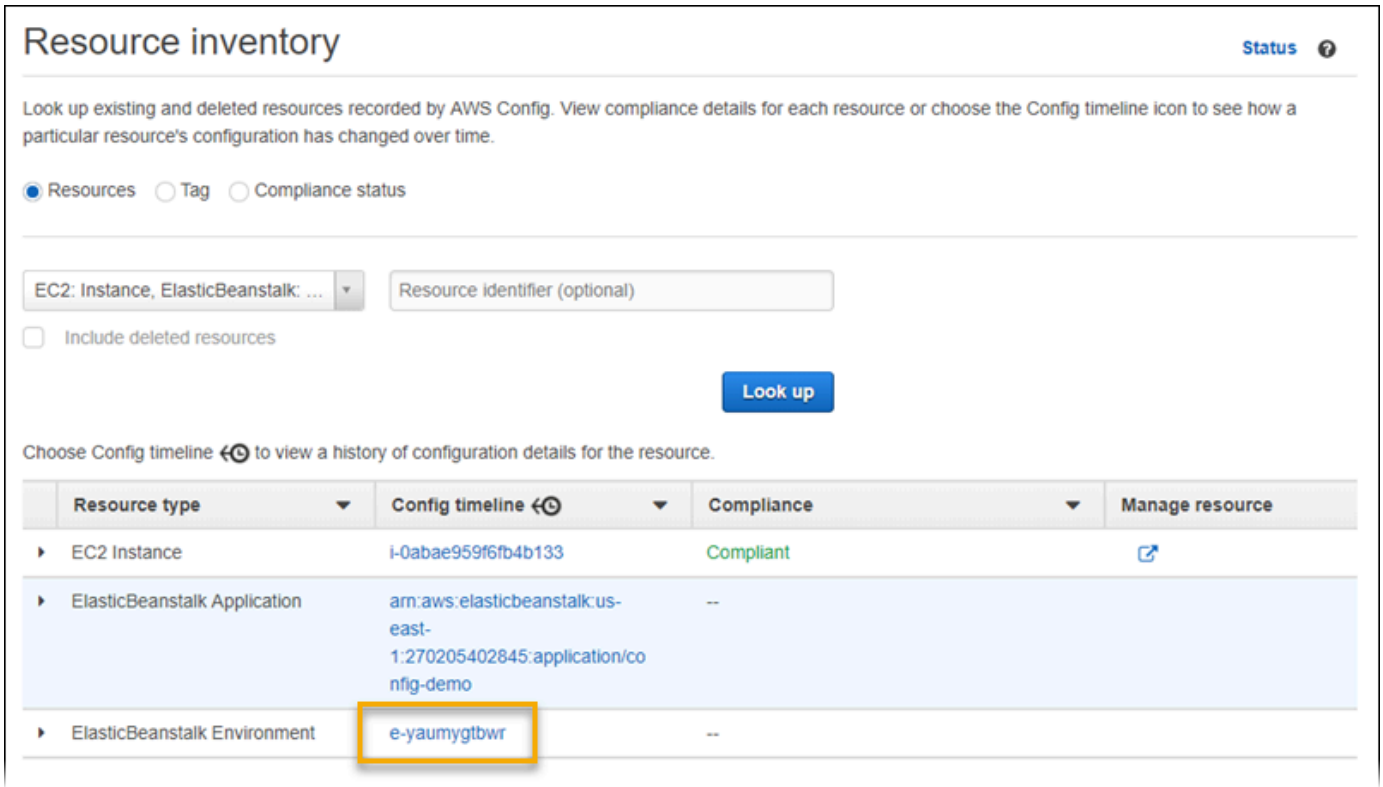
1. 打开 AWS Config 控制台。
2. 选择 Resources (资源)。
3. 在 Resource (资源) 清单页面上，选择 Resources (资源)。
4. 打开 Resource type (资源类型) 菜单，滚动到 ElasticBeanstalk，然后选择一个或多个 Elastic Beanstalk 资源类型。

   > **Note**
   >
   > 要查看 Elastic Beanstalk 为您的应用程序创建的其他资源的配置详细信息，请选择更多资源类型。例如，您可以选择 EC2 下的 Instance (实例)。

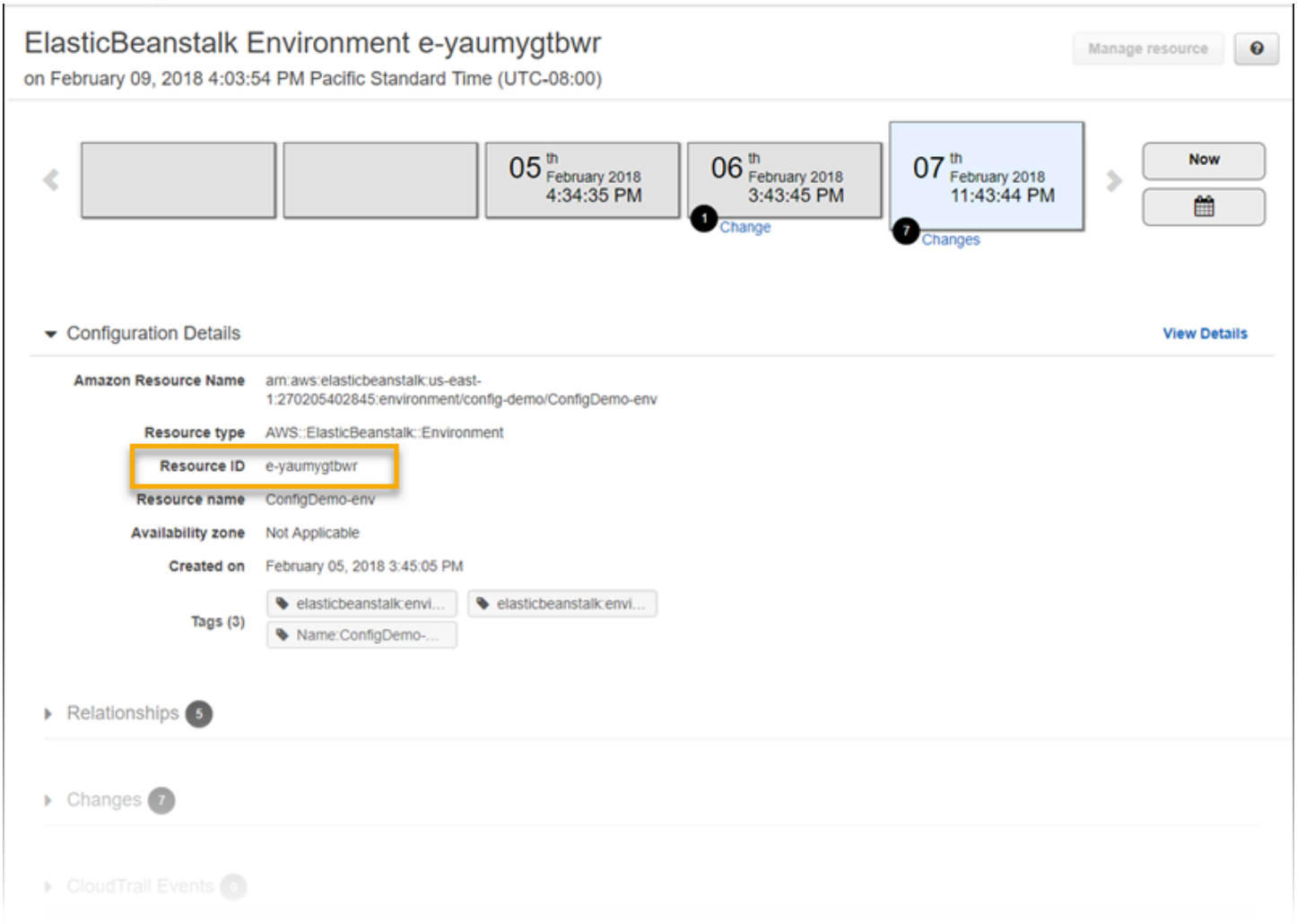
5. 选择 Look up (查找)。参阅下图中的 2。



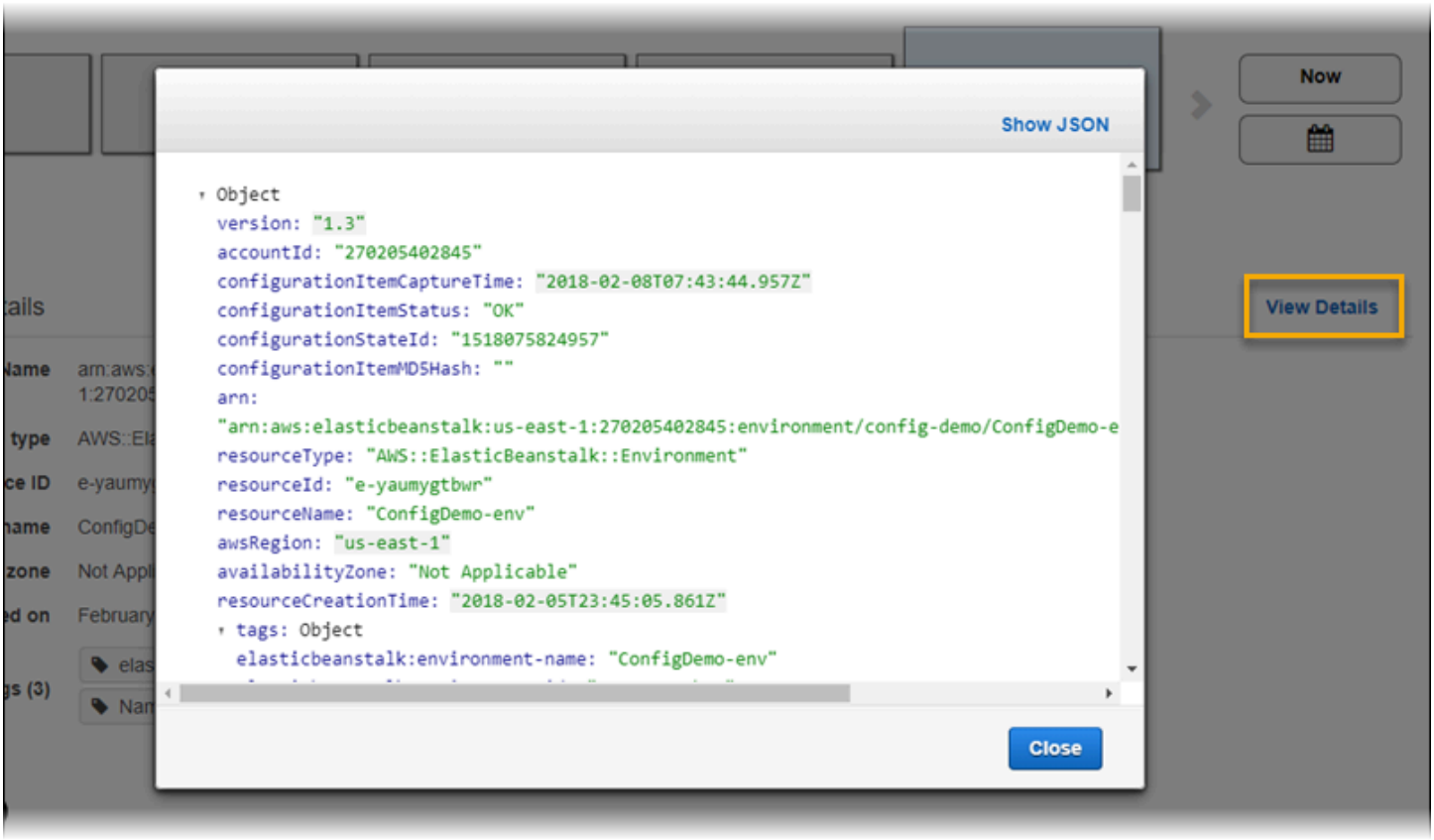
6. 在 AWS Config 显示的资源列表中选择一个资源 ID。



AWS Config 显示了有关您选择的资源的配置详细信息和其他信息。



要查看记录的配置的完整详细信息，请选择 View Details (查看详细信息)。



要了解在此页面上查找资源和查看信息的更多方法，请于 AWS 开发人员指南中参阅查看“AWS Config 资源配置和历史记录”。

使用 AWS Config 规则评估 Elastic Beanstalk 资源

您可以创建 AWS Config 规则，这些规则表示 Elastic Beanstalk 资源的理想配置设置。您可以使用预定义的 AWS 托管 Config 规则，也可以定义自定义规则。AWS Config 会持续跟踪对您的资源配置的更改，以确定这些更改是否符合规则中设定的所有条件。AWS Config 控制台将显示您的规则与资源的合规性状态。

如果某个资源违反了某规则并且被标记为 noncompliant (不合规)，则 AWS Config 可能提醒您使用 Amazon Simple Notification Service (Amazon SNS) 主题。要以编程方式使用这些 AWS Config 提醒中的数据，请使用 Amazon Simple Queue Service (Amazon SQS) 队列作为 Amazon SNS 主题的通知终端节点。例如，您可能希望编写代码，以便在有人修改您环境的 Auto Scaling 组配置时启动工作流。

要了解有关设置和使用规则的更多信息，请参阅 AWS Config 开发人员指南中的使用 AWS Config 规则评估资源。

配合使用 Elastic Beanstalk 和 Amazon DynamoDB

Amazon DynamoDB 是一种完全托管的 NoSQL 数据库服务，提供快速而可预测的性能，能够实现无缝扩展。如果您是一名开发人员，您可以使用 DynamoDB 创建一个数据库表来存储和检索任意数量的数据，并处理任何级别的请求流量。DynamoDB 自动将表的数据和流量分布到足够数量的服务器上，以处理客户指定的请求容量和存储的数据量，同时保持一致且快速的性能。所有数据项目均存储在固态硬盘 (SSD) 中，并自动复制到某个 AWS 区域的多个可用区，以便提供数据自身的高可用性和数据持久性。

如果您在工作线程环境中使用定期任务，Elastic Beanstalk 将创建一个 DynamoDB 表，并使用此表执行领导选择和存储有关任务的信息。环境中的每个实例均会每隔几秒就尝试对表进行一次写入以变为领导，并按计划执行任务。

您可以使用配置文件为应用程序创建 DynamoDB 表。有关使用配置文件创建表并使用 AWS SDK for JavaScript in Node.js 连接到此表的示例 Node.js 应用程序，请参阅 GitHub 上的 eb-node-express-sample。有关将 DynamoDB 与 PHP 结合使用的示例演练，请参阅示例：DynamoDB、CloudWatch 和 SNS。有关使用 AWS SDK for Java 的示例，请参阅 AWS SDK for Java 文档中的使用 DynamoDB 管理 Tomcat 会话状态。

当您使用配置文件创建 DynamoDB 表时，该表不会与环境的生命周期关联，也不会在您终止环境时删除该表。要确保个人信息不会被不必要地保留，请删除不再需要的任何记录或删除该表。

有关 DynamoDB 的更多信息，请参阅 DynamoDB 开发人员指南。

将 Elastic Beanstalk 和 Amazon ElastiCache 结合使用

Amazon ElastiCache 是一项 Web 服务，可在云中设置、管理和扩展分布式内存中的缓存环境。它可以提供高性能、可扩展且符合成本效益的内存缓存，同时消除部署和管理分布式缓存环境产生的相关复杂性。ElastiCache 的协议符合 Redis 和 Memcached，因此您用于现有 Redis 和 Memcached 环境的代码、应用程序和大多数流行工具可无缝应用于本服务。有关 ElastiCache 的更多信息，请转到 Amazon ElastiCache 产品页。

将 Elastic Beanstalk 和 Amazon ElastiCache 结合使用

1. 创建 ElastiCache 群集。

   - 有关如何使用 Redis 创建 ElastiCache 集群的说明，请转至 ElastiCache for Redis 用户指南 中的 Amazon ElastiCache for Redis 入门。

- 有关如何使用 Memcached 创建 ElastiCache 集群的说明，请转至 ElastiCache for Memcached 用户指南 中的 Amazon ElastiCache for Memcached 入门。

2. 配置 ElastiCache 安全组，以便获得从 Elastic Beanstalk 应用程序所用的 Amazon EC2 安全组中进行访问的权限。有关如何使用AWS管理控制台查找 EC2 安全组名称的说明，请参阅 EC2 实例文档页上的安全组。

   - 有关 Redis 的更多信息，请转到 ElastiCache for Redis 用户指南 中的授权访问。
   - 有关 Memcached 的更多信息，请转到 ElastiCache for Memcached 用户指南 中的授权访问。

您可以使用配置文件对您的 Elastic Beanstalk 环境进行自定义，以便使用 ElastiCache。有关将 ElastiCache 与 Elastic Beanstalk 集成的配置文件示例，请参阅示例：ElastiCache。

配合使用 Elastic Beanstalk 和 Amazon Elastic File System

借助 Amazon Elastic File System (Amazon EFS)，您可以创建可由跨多个可用区的实例装载的网络文件系统。Amazon EFS 文件系统是一种 AWS 资源，它使用安全组来控制默认或自定义 VPC 中的网络访问。

在 Elastic Beanstalk 环境中，您可以使用 Amazon EFS 创建共享目录，以存储用户为应用程序上传或修改的文件。您的应用程序可以处理已装载的 Amazon EFS 卷，例如本地存储。这样，您不必更改应用程序代码即可纵向扩展到多个实例。

有关 Amazon EFS 的更多信息，请参阅 Amazon Elastic File System 用户指南。

> **Note**
>
> Elastic Beanstalk 创建了 webapp 用户，您能够以应用程序目录所有者的身份在 Amazon EC2 实例上对其进行设置。有关更多信息，请参阅本指南设计考虑因素主题中的持久性存储。

小节目录

配置文件

Elastic Beanstalk 提供了可用于创建和装载 Amazon EFS 文件系统的配置文件。您可以创建 Amazon EFS 卷作为环境的一部分，也可以装载独立于 Elastic Beanstalk 创建的 Amazon EFS 卷。

- storage-efs-createfilesystem.config – 使用 `Resources` 键在 Amazon EFS 中创建新的文件系统和装载点。您的环境中的所有实例都可以连接同一个文件系统，以实现共享、可扩展的存储。使用 `storage-efs-mountfilesystem.config` 在每个实例上装载文件系统。

  > **内部资源**
  >
  > 使用配置文件创建的任何资源都与环境的生命周期关联。如果终止环境或删除配置文件，这些资源将丢失。

- storage-efs-mountfilesystem.config – 将 Amazon EFS 文件系统装载到环境中实例上的本地路径。您可以使用 `storage-efs-createfilesystem.config` 创建卷作为环境的一部分。或者，您可以使用 Amazon EFS 控制台、AWS CLI 或 AWS SDK 将其挂载到您的环境。

要使用配置文件，请先使用 `storage-efs-createfilesystem.config` 创建 Amazon EFS 文件系统。按照配置文件中的说明将其添加到源代码中的 .ebextensions 目录里，以便在 VPC 中创建此文件系统。

将更新后的源代码部署到 Elastic Beanstalk 环境。这是为了确认文件系统已成功创建。然后，添加 `storage-efs-mountfilesystem.config` 以将此文件系统装载到环境中的实例上。在两个独立的部署过程中执行此操作可确保即使挂载操作失败，文件系统仍保持完整。如果在同一部署中同时执行两个步骤，则任何一个步骤出现问题都将导致文件系统在部署失败时终止。

加密文件系统

Amazon EFS 支持加密的文件系统。本主题中讨论的 `storage-efs-createfilesystem.config` 配置文件定义了两个自定义选项。您可以使用这些选项创建 Amazon EFS 加密文件系统。有关更多信息，请参阅配置文件中的说明。

示例应用程序

Elastic Beanstalk 还提供了使用 Amazon EFS 作为共享存储的示例应用程序。这两个项目都具有配置文件，您可以将其与标准的 WordPress 或 Drupal 安装程序配合使用，以在负载均衡环境中运行博客或其他内容管理系统。当用户上传照片或其他媒体时，该文件将存储在 Amazon EFS 文件系统中。这避免了必须使用替代方案，即使用插件在 Amazon S3 中存储上传的文件。

- 负载均衡的 WordPress – 这包括用于安全安装 WordPress 并在负载均衡的 Elastic Beanstalk 环境中运行 WordPress 的配置文件。
- 负载均衡的 Drupal – 这包括用于安全安装 Drupal 并在负载均衡的 Elastic Beanstalk 环境中运行 Drupal 的配置文件和说明。

清除文件系统

如果您已创建 Amazon EFS 文件系统，以使用配置文件作为 Elastic Beanstalk 环境的一部分，则 Elastic Beanstalk 会在您终止环境时删除该文件系统。为最大限度降低运行应用程序的存储成本，请定期删除应用程序不需要的文件。或者，确保应用程序代码正确维护文件生命周期。

> **Important**
>
> 如果您已在 Elastic Beanstalk 环境外部创建 Amazon EFS 文件系统并已将该系统装载到环境的实例中，则 Elastic Beanstalk 不会在您终止环境时删除该文件系统。要确保不会保留个人信息且避免存储成本，请在您不再需要应用程序所存储的文件时删除它们。或者，您也可以删除整个文件系统。

将 Elastic Beanstalk 与 AWS Identity and Access Management

AWS Identity and Access Management (IAM) 可帮助您安全地控制对 AWS 资源的访问权限。本部分包含了有关使用 IAM 策略、实例配置文件和服务角色的参考材料。

有关权限的概述，请参阅服务角色、实例配置文件和用户策略。对于大多数环境，在您启动第一个环境时 Elastic Beanstalk 控制台提示您创建的服务角色和实例配置文件具有所需的所有权限。同样，Elastic Beanstalk 提供的用于完全访问和只读访问的托管式策略包含日常使用所需的所有用户权限。

I AM 用户指南对 AWS 权限进行了深入介绍。

主题

- 管理 Elastic Beanstalk 实例配置文件
- 管理 Elastic Beanstalk 服务角色
- 将服务相关角色用于 Elastic Beanstalk
- 管理 Elastic Beanstalk 用户策略

- Elastic Beanstalk 的 Amazon 资源名称格式
- Elastic Beanstalk 操作的资源和条件
- 使用标签控制对 Elastic Beanstalk 资源的访问
- 基于托管式策略的示例策略
- 基于资源权限的示例策略
- 防止跨环境访问 Amazon S3 存储桶

管理 Elastic Beanstalk 实例配置文件

实例配置文件是 AWS Identity and Access Management (IAM) 角色的容器，您可以使用该容器在实例启动时将角色信息传递给 Amazon EC2 实例。

如果您的 AWS 账户没有 EC2 实例配置文件，则必须使用 IAM 服务创建一个。然后，您可以将此 EC2 实例配置文件分配到您创建的新环境。创建环境向导提供了相关的信息，以在您使用 IAM 服务过程提供指导，确保您可以创建具有所需权限的 EC2 实例配置文件。创建该实例配置文件后，您可以返回控制台以将其选中并作为 EC2 实例配置文件，然后继续执行创建环境的相关步骤。

> **Note**
>
> 以前，Elastic Beanstalk 创建了一个默认 EC2 实例`aws-elasticbeanstalk-ec2-role`配置文件，该配置文件名为账户首次创建环境 AWS 时命名。该实例配置文件包含默认的托管式策略。如果您的账户已经有该实例配置文件，则可继续将其分配到您的环境。
> 但是，最近的 AWS 安全准则不允许 AWS 服务自动创建具有对其他 AWS 服务（在本例中为 EC2）的信任策略的角色。根据这些安全准则，Elastic Beanstalk 将不再创建默认的 `aws-elasticbeanstalk-ec2-role` 实例配置文件。

托管策略

Elastic Beanstalk 提供多种托管式策略，以确保您的环境能够满足不同的使用场景需求。要满足环境的默认使用场景需要，必须将这些策略附加到 EC2 实例配置文件对应的角色。

- AWSElasticBeanstalkWebTier— 授予应用程序将日志上传到 Amazon S3 并将调试信息上传到的权限 AWS X-Ray。要查看托管策略的内容，请参阅AWSElasticBeanstalkWebTier《AWS 托管策略参考指南》。

- AWSElasticBeanstalkWorkerTier— 授予日志上传、调试、指标发布和工作器实例任务（包括队列管理、领导者选举和定期任务）的权限。要查看托管策略的内容，请参阅AWSElasticBeanstalkWorkerTier《AWS 托管策略参考指南》。
- AWSElasticBeanstalkMulticontainerDocker— 向 Amazon 弹性容器服务授予权限，以协调 Docker 环境的集群任务。要查看托管策略的内容，请参阅AWSElasticBeanstalkMulticontainerDocker《AWS 托管策略参考指南》。

> **Important**
>
> Elastic Beanstalk 托管式策略不提供精细权限—它们授予使用 Elastic Beanstalk 应用程序可能需要的所有权限。在某些情况下，您可能希望进一步限制我们的托管策略的权限。有关一个用例的示例，请参阅防止跨环境访问 Amazon S3 存储桶。
> 我们的托管式策略也不涵盖对您可能添加到解决方案中且不由 Elastic Beanstalk 管理的自定义资源的权限。要实施更精细的权限、所需的最低权限或自定义资源权限，请使用自定义策略。

EC2 的信任关系政策

要允许环境中的 EC2 实例代入所需的角色，实例配置文件必须将 Amazon EC2 指定为信任关系策略中的可信实体，如下所示。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

要自定义权限，您可以向附加到默认实例配置文件的角色添加策略，或者创建具有一组受限权限的实例配置文件。

Sections

创建实例配置文件

实例配置文件是一个面向标准 IAM 角色的包装程序，它允许 EC2 实例代入该角色。您可以创建其他实例配置文件来为不同的应用程序自定义权限。或者，如果您不使用 Worker 层或 ECS 托管式 Docker 环境，则可以创建不向这些功能授予权限的实例配置文件。

要创建实例配置文件

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择创建角色。
3. 在可信实体类型下，选择 AWS 服务。
4. 在 Use case（使用案例）下，选择 EC2。
5. 选择下一步。
6. 附加由 Elastic Beanstalk 提供的适当托管式策略以及提供应用程序所需的权限的任何其他策略。
7. 选择下一步。
8. 输入角色的名称。
9. （可选）将标签添加到角色。
10. 选择 创建角色。

验证分配给实例配置文件的权限

分配给您的默认实例配置文件的权限不是固定的，具体取决于其创建时间、您上次启动环境的时间和您使用的客户端。您可以在 IAM 控制台中验证默认实例配置文件的权限。

验证默认实例配置文件的权限

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择分配为 EC2 实例配置文件的角色。
3. 在 Permissions（权限）选项卡中，审核附加到角色的策略列表。
4. 要查看策略授予的权限，请选择相应的策略。

更新 out-of-date 默认实例配置文件

如果默认实例配置文件缺少所需的权限，您可以手动将托管式策略附加到分配为 EC2 实例配置文件的角色。

向附加到默认实例配置文件的角色添加托管式策略

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择分配为 EC2 实例配置文件的角色。
3. 在 Permissions（权限）选项卡上，选择 Attach policies（附加策略）。
4. 键入 **AWSElasticBeanstalk** 以筛选策略。
5. 选择下列策略，然后选择 Attach policy（附加策略）：

   - `AWSElasticBeanstalkWebTier`
   - `AWSElasticBeanstalkWorkerTier`
   - `AWSElasticBeanstalkMulticontainerDocker`

向默认实例配置文件添加权限

如果您的应用程序访问默认实例配置文件中未授予权限 AWS 的 API 或资源，请在 IAM 控制台中添加授予权限的策略。

向附加到默认实例配置文件的角色添加策略

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择分配为 EC2 实例配置文件的角色。
3. 在 Permissions（权限）选项卡上，选择 Attach policies（附加策略）。
4. 选择适用于应用程序使用的附加服务的托管策略。例如，`AmazonS3FullAccess` 或 `AmazonDynamoDBFullAccess`。
5. 选择 Attach policy（附加策略）。

管理 Elastic Beanstalk 服务角色

要管理和监控您的环境，请代表您对环境资源 AWS Elastic Beanstalk 执行操作。Elastic Beanstalk 需要一定的权限才能执行这些操作， AWS Identity and Access Management 它假设 (IAM) 服务角色才能获得这些权限。

Elastic Beanstalk 在其代入服务角色时需要使用临时安全凭证。要获取这些凭证，Elastic Beanstalk 会将请求发送到一个特定于区域的端点上的 AWS Security Token Service（AWS STS）。有关更多信息，请参阅《IAM 用户指南》中的临时安全凭证。

> **Note**
>
> 如果您的环境所在区域的 AWS STS 终端节点被停用，Elastic Beanstalk 会将请求发送到无法停用的备用终端节点。此端点与不同的区域关联。因此，该请求是跨区域请求。有关更多信息，请参阅 IAM 用户指南 AWS STS 中的在 AWS 区域激活和停用。

使用 Elastic Beanstalk 控制台和 EB CLI 管理服务角色

您可以使用 Elastic Beanstalk 控制台和 EB CLI 为环境设置具有足够权限集的服务角色。它们创建默认服务角色并在其中使用托管式策略。

托管服务角色策略

Elastic Beanstalk 提供一个用于增强型运行状况监控的托管式策略和一个具有托管平台更新所需的附加权限的托管式策略。控制台和 EB CLI 将这两个策略分配到它们为您创建的默认服务角色。这些策略仅应当用于此默认服务角色。它们不应与您账户中的其他用户或角色一起使用。

`AWSElasticBeanstalkEnhancedHealth`

此策略向 Elastic Beanstalk 授予监控实例和环境运行状况的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetHealth",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:GetConsoleOutput",
                "ec2:AssociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeSecurityGroups",
                "sqs:GetQueueAttributes",
```

```
                "sqs:GetQueueUrl",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeNotificationConfigurations",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy

此策略向 Elastic Beanstalk 授予权限，以代表您更新环境以执行托管平台更新。

服务级别权限分组

此策略根据提供的权限集分为多个语句。

- *ElasticBeanstalkPermissions* – 这一组权限用于调用 Elastic Beanstalk 服务操作（Elastic Beanstalk API）。
- *AllowPassRoleToElasticBeanstalkAndDownstreamServices* – 这一组权限允许将任何角色传递给 Elastic Beanstalk 及其他下游服务，例如 AWS CloudFormation。
- *ReadOnlyPermissions* – 这一组权限用于收集有关运行环境的信息。
- **OperationPermissions* – 采用此命名模式的组用于调用执行平台更新必需的操作。
- **BroadOperationPermissions* – 采用此命名模式的组用于调用执行平台更新必需的操作。它们还包括支持旧环境的广泛权限。
- **TagResource*— 使用这种命名模式的群组适用于使用 tag-on-create API 为在 Elastic Beanstalk 环境中创建的资源附加标签的调用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ElasticBeanstalkPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "elasticbeanstalk:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowPassRoleToElasticBeanstalkAndDownstreamServices",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "elasticbeanstalk.amazonaws.com",
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "autoscaling.amazonaws.com",
                        "elasticloadbalancing.amazonaws.com",
                        "ecs.amazonaws.com",
                        "cloudformation.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "ReadOnlyPermissions",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeLoadBalancers",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeScheduledActions",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstances",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
```

```
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "logs:DescribeLogGroups",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeOrderableDBInstanceOptions",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2BroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:DeleteSecurityGroup",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2RunInstancesOperationPermissions",
            "Effect": "Allow",
```

```
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "ec2:LaunchTemplate": "arn:aws:ec2:*:*:launch-template/*"
                }
            }
        },
        {
            "Sid": "EC2TerminateInstancesOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-id": [
                        "arn:aws:cloudformation:*:*:stack/awseb-e-*",
                        "arn:aws:cloudformation:*:*:stack/eb-*"
                    ]
                }
            }
        },
        {
            "Sid": "ECSBroadOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:DescribeClusters",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ECSDeleteClusterOperationPermissions",
            "Effect": "Allow",
            "Action": "ecs:DeleteCluster",
            "Resource": "arn:aws:ecs:*:*:cluster/awseb-*"
        },
        {
            "Sid": "ASGOperationPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "autoscaling:AttachInstances",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteScheduledAction",
                "autoscaling:DetachInstances",
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
                "autoscaling:PutScheduledUpdateGroupAction",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:ResumeProcesses",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:SuspendProcesses",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": [
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
awseb-e-*",
                "arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/
eb-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/awseb-
e-*",
                "arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/eb-*"
            ]
        },
        {
            "Sid": "CFNOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        },
        {
            "Sid": "ELBOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
```

```
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:*:*:targetgroup/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:targetgroup/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/awseb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/eb-*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/awseb-*/*",
                "arn:aws:elasticloadbalancing:*:*:loadbalancer/*/eb-*/*"
            ]
        },
        {
            "Sid": "CWLogsOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/elasticbeanstalk/*"
        },
        {
            "Sid": "S3ObjectOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectVersionAcl"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*/*"
        },
        {
```

```
            "Sid": "S3BucketOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucket",
                "s3:PutBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::elasticbeanstalk-*"
        },
        {
            "Sid": "SNSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:GetTopicAttributes",
                "sns:SetTopicAttributes",
                "sns:Subscribe"
            ],
            "Resource": "arn:aws:sns:*:*:ElasticBeanstalkNotifications-*"
        },
        {
            "Sid": "SQSOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl"
            ],
            "Resource": [
                "arn:aws:sqs:*:*:awseb-e-*",
                "arn:aws:sqs:*:*:eb-*"
            ]
        },
        {
            "Sid": "CWPutMetricAlarmOperationPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:awseb-*",
                "arn:aws:cloudwatch:*:*:alarm:eb-*"
            ]
        },
```

```
        {
            "Sid": "AllowECSTagResource",
            "Effect": "Allow",
            "Action": [
                "ecs:TagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterTaskDefinition"
                    ]
                }
            }
        }
    ]
}
```

要查看托管式策略的内容，您还可以使用 IAM 控制台中的 Policies（策略）页面。

> **Important**
>
> Elastic Beanstalk 托管式策略不提供精细权限—它们授予使用 Elastic Beanstalk 应用程序可能需要的所有权限。在某些情况下，您可能希望进一步限制我们的托管策略的权限。有关一个用例的示例，请参阅防止跨环境访问 Amazon S3 存储桶。
> 我们的托管式策略也不涵盖对您可能添加到解决方案中且不由 Elastic Beanstalk 管理的自定义资源的权限。要实施更精细的权限、所需的最低权限或自定义资源权限，请使用自定义策略。

已弃用的 托管策略

过去，Elastic Beanstalk AWSElasticBeanstalkService支持托管服务角色策略。此政策已被替换为AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy。您可能仍然能够在 IAM 控制台中查看和使用较早的策略。

要查看托管策略的内容，请参阅AWSElasticBeanstalkService《AWS 托管策略参考指南》。

但是，我们建议您过渡到使用新的托管策略(AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy)。添加自定义策略以授予对自定义资源（如果有）的权限。

使用 Elastic Beanstalk 控制台

在 Elastic Beanstalk 控制台中启动环境时，该控制台创建一个名为 `aws-elasticbeanstalk-service-role` 的默认服务角色，并将具有默认权限的托管式策略附加到该服务角色。

为了允许 Elastic Beanstalk 代入 `aws-elasticbeanstalk-service-role` 角色，该服务角色在信任关系策略中将 Elastic Beanstalk 指定为受信任实体。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": {
          "Service": "elasticbeanstalk.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
            "sts:ExternalId": "elasticbeanstalk"
          }
        }
      }
    ]
}
```

当您为环境启用托管平台更新时，Elastic Beanstalk 会代入单独的托管更新服务角色以执行托管更新。默认情况下，Elastic Beanstalk 控制台会为托管更新服务角色使用生成的相同服务角色 `aws-elasticbeanstalk-service-role`。如果您更改默认服务角色，控制台将设置托管更新服务角色，以使用托管更新服务相关角色 `AWSServiceRoleForElasticBeanstalkManagedUpdates`。有关服务相关角色的更多信息，请参阅the section called “使用服务相关角色”。

> **Note**
>
> 由于权限问题，Elastic Beanstalk 服务并不总是能成功为您创建此服务相关角色。因此，控制台尝试明确创建它。要确保您的账户具有此服务相关角色，请使用控制台至少创建一个环境，并在创建环境之前配置启用了托管更新。

使用 EB CLI

如果您使用 Elastic Beanstalk 命令行界面（EB CLI）的 the section called “eb create” 命令启动环境但未通过 `--service-role` 选项指定服务角色，Elastic Beanstalk 将创建默认服务角色 `aws-elasticbeanstalk-service-role`。如果默认服务角色已存在，Elastic Beanstalk 会将其用于新环境。在这些情况下，Elastic Beanstalk 控制台也会执行类似的操作。

与控制台不同，当您使用 EB CLI 命令选项时您不能指定托管更新服务角色。如果您为环境启用了托管更新，您必须通过配置选项设置托管更新服务角色。以下示例启用托管更新并使用默认服务角色作为托管更新服务角色。

Example .ebextensions/ .config managed-platform-update

```
option_settings:
  aws:elasticbeanstalk:managedactions:
    ManagedActionsEnabled: true
    PreferredStartTime: "Tue:09:00"
    ServiceRoleForManagedUpdates: "aws-elasticbeanstalk-service-role"
  aws:elasticbeanstalk:managedactions:platformupdate:
    UpdateLevel: patch
    InstanceRefreshEnabled: true
```

使用 Elastic Beanstalk API 管理服务角色

当您使用 Elastic Beanstalk API 的 `CreateEnvironment` 操作来创建环境时，请使用 `aws:elasticbeanstalk:environment` 命名空间中的 `ServiceRole` 配置选项指定服务角色。有关在 Elastic Beanstalk API 中使用增强型运行状况监控的详细信息，请参阅将增强型运行状况报告与 Elastic Beanstalk API 结合使用。

此外，如果您为环境启用托管平台更新，则可以使用 `aws:elasticbeanstalk:managedactions` 命名空间的 `ServiceRoleForManagedUpdates` 选项指定托管更新服务角色。

使用服务相关角色

服务相关角色是一种独特的服务角色，由 Elastic Beanstalk 预定义，包括该服务代表您调用其他服务所需的所有权限。 AWS 服务相关角色与您的账户关联。Elastic Beanstalk 将创建一次，然后在创建其他环境时重用。有关将服务相关角色与 Elastic Beanstalk 环境结合使用的详细信息，请参阅将服务相关角色用于 Elastic Beanstalk。

如果您使用 Elastic Beanstalk API 创建环境并且未指定服务角色，Elastic Beanstalk 将为您的账户创建监控服务相关角色（如果尚不存在该角色）。Elastic Beanstalk 将此角色用于新环境。您也可以使

用 IAM 提前为账户创建监控服务相关角色。在您的账户拥有此角色后，您可以通过 Elastic Beanstalk API、Elastic Beanstalk 控制台或 EB CLI 使用该角色创建环境。

如果您为环境启用托管式平台更新并指定 `AWSServiceRoleForElasticBeanstalkManagedUpdates` 作为 `aws:elasticbeanstalk:managedactions` 命名空间 `ServiceRoleForManagedUpdates` 选项的值，则 Elastic Beanstalk 会为您的账户创建托管式更新服务相关角色（如果该角色尚不存在）。Elastic Beanstalk 使用该角色为新环境执行托管式更新。

> **Note**
>
> 在您创建环境时，如果 Elastic Beanstalk 尝试为您的账户创建监控和托管更新服务相关角色，您必须具有 `iam:CreateServiceLinkedRole` 权限。如果您没有此权限，环境创建将失败，并显示说明问题的消息。
> 作为替代方案，也可以让具有创建服务相关角色权限的另一用户使用 IAM 事先创建服务相关角色。使用此方法，您不需要 `iam:CreateServiceLinkedRole` 权限即可创建环境。

验证默认服务角色权限

您的默认服务角色授予的权限不固定，具体取决于其创建时间、您上次启动环境的时间以及您使用的客户端。在 IAM 控制台中，您可以验证由默认服务角色授予的权限。

验证默认服务角色的权限

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择aws-elasticbeanstalk-service-role。
3. 在 Permissions（权限）选项卡中，审核附加到角色的策略列表。
4. 要查看策略授予的权限，请选择相应的策略。

更新 out-of-date 默认服务角色

如果默认服务角色缺少必需的权限，您可以通过在 Elastic Beanstalk 环境管理控制台中创建新环境来更新它。

或者，您可以手动向默认服务角色添加托管式策略。

向默认服务角色添加托管式策略

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择aws-elasticbeanstalk-service-role。
3. 在 Permissions（权限）选项卡上，选择 Attach policies（附加策略）。
4. 输入 **AWSElasticBeanstalk** 以筛选策略。
5. 选择下列策略，然后选择 Attach policy（附加策略）：
   - `AWSElasticBeanstalkEnhancedHealth`
   - `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy`

向默认服务角色添加权限

如果您的应用程序包含引用默认服务角色中未包含权限的 AWS 资源的配置文件，则 Elastic Beanstalk 可能需要额外的权限。在托管更新期间处理配置文件时，需要这些附加权限才能解析这些引用。如果缺少权限，则更新将失败，并且 Elastic Beanstalk 将返回一条消息来指明所需的权限。按照这些步骤，在 IAM 控制台中将附加服务的权限添加到默认服务角色。

向默认服务角色添加附加策略

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择aws-elasticbeanstalk-service-role。
3. 在 Permissions（权限）选项卡上，选择 Attach policies（附加策略）。
4. 选择适用于应用程序使用的附加服务的托管策略。例如，`AmazonAPIGatewayAdministrator` 或 `AmazonElasticFileSystemFullAccess`。
5. 选择 Attach policy（附加策略）。

创建服务角色

如果您无法使用默认服务角色，请创建一个服务角色。

创建服务角色

1. 在 IAM 控制台中，打开 Roles（角色）页面。
2. 选择 创建角色。

3. 在 AWS service（亚马逊云科技服务）下，选择 AWS Elastic Beanstalk，然后选择您的使用案例。
4. 选择下一步: 权限。
5. 附加 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 和 `AWSElasticBeanstalkEnhancedHealth` 托管式策略以及提供应用程序所需权限的任何其他策略。
6. 选择 Next: Tags（下一步: 标签）。
7. （可选）将标签添加到角色。
8. 选择 Next: Review（下一步: 审核）。
9. 输入角色的名称。
10. 选择 Create role（创建角色）。

在使用环境创建向导或通过 `eb create` 命令中的 `--service-role` 选项创建环境时应用您的自定义服务角色。

将服务相关角色用于 Elastic Beanstalk

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与 Elastic Beanstalk 直接关联的独特类型的 IAM 角色。服务相关角色由 Elastic Beanstalk 预定义，包括该服务代表您调用其他服务所需的所有权限。 AWS

Elastic Beanstalk 定义了几种类型的服务相关角色：

- 监控服务相关角色 – 允许 Elastic Beanstalk 监控正在运行的环境的运行状况并发布运行状况事件通知。
- 维护服务相关角色 – 允许 Elastic Beanstalk 对您正在运行的环境执行定期维护活动。
- 托管更新服务相关角色 – 允许 Elastic Beanstalk 对您正在运行的环境执行计划的平台更新。

主题

监控服务相关角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与 Elastic Beanstalk 直接关联的独特类型的 IAM 角色。服务相关角色由 Elastic Beanstalk 预定义，包括该服务代表您调用其他服务所需的所有权限。 AWS

您可以使用服务相关角色轻松设置 Elastic Beanstalk，因为您不必手动添加所需的权限。Elastic Beanstalk 定义了其服务相关角色的权限，除非另外定义，否则只有 Elastic Beanstalk 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 Elastic Beanstalk 资源，因为您无法无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的AWS 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。选择 Yes（是）和链接，查看该服务的服务相关角色文档。

Elastic Beanstalk 的服务相关角色权限

Elastic Beanstalk 使用名为的服务相关角色 AWSServiceRoleForElasticBeanstalk——允许 Elastic Beanstalk 监控运行环境的运行状况并发布运行状况事件通知。

AWSServiceRoleForElasticBeanstalk 服务相关角色信任以下服务来代入该角色：

- `elasticbeanstalk.amazonaws.com`

AWSServiceRoleForElasticBeanstalk 服务相关角色的权限策略包含 Elastic Beanstalk 代表您完成操作所需的所有权限：

AllowCloudformationReadOperationsOnElasticBeanstalkStacks

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudformationReadOperationsOnElasticBeanstalkStacks",
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks"
            ],
```

```
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        },
        {
            "Sid": "AllowOperations",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:PutNotificationConfiguration",
                "ec2:DescribeInstanceStatus",
                "ec2:AssociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeInstances",
                "ec2:DescribeSecurityGroups",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTargetGroups",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

或者，您可以使用 AWS 托管策略提供对 Elastic Beanstalk 的完全访问权限。

为 Elastic Beanstalk 创建服务相关角色

您无需手动创建服务相关角色。当您使用 Elastic Beanstalk API 创建 Elastic Beanstalk 环境但未指定服务角色时，Elastic Beanstalk 会为您创建服务相关角色。

> **Important**
>
> 如果您在 2017 年 9 月 27 日开始 AWSServiceRoleForElasticBeanstalk 支持服务相关角色之前使用过 Elastic Beanstalk 服务，而您的账户需要该服务，那么 Elastic Beanstalk 会在您的账户中创建该角色。 AWSServiceRoleForElasticBeanstalk 要了解更多信息，请参阅我的 IAM 账户中的新角色。

当 Elastic Beanstalk 在 AWSServiceRoleForElasticBeanstalk 创建环境时尝试为您的账户创建服务相关角色时，您必须拥有该权限。`iam:CreateServiceLinkedRole`如果您没有此权限，环境创建将失败，您会看到说明问题的消息。

作为替代方案，也可以让具有创建服务相关角色权限的另一用户使用 IAM 预先创建服务相关角色。这样，即使您没有 `iam:CreateServiceLinkedRole` 权限，也可以创建环境。

您（或其他用户）可以使用 IAM 控制台，通过 Elastic Beanstalk 使用案例创建服务相关角色。在 IAM CLI 或 IAM API 中，用 `elasticbeanstalk.amazonaws.com` 服务名称创建一个服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您使用 Elastic Beanstalk API 创建 Elastic Beanstalk 环境但未指定服务角色时，Elastic Beanstalk 会再次为您创建服务相关角色。

为 Elastic Beanstalk 编辑服务相关角色

Elastic Beanstalk 不允许你编辑服务相关角色。 AWSServiceRoleForElasticBeanstalk 创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参见《IAM 用户指南》中的编辑服务相关角色。

删除 Elastic Beanstalk 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

清除服务相关角色

您必须首先确保所有 Elastic Beanstalk 环境使用的是不同服务角色或已终止，然后才能使用 IAM 删除服务相关角色。

> **Note**
>
> 在您尝试终止环境时，如果 Elastic Beanstalk 服务在使用服务相关角色，终止操作可能会失败。如果发生这种情况，请等待几分钟后重试。

终止使用（控制台）的 Elastic Beanstalk 环境 AWSServiceRoleForElasticBeanstalk

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

有关使用 EB CLI 终止 Elastic Beanstalk 环境的详细信息，请参阅eb terminate。

有关TerminateEnvironment使用 API 终止 Elastic Beanstalk 环境的详细信息，请参阅。

手动删除服务相关角色

使用 IAM 控制台、IAM CLI 或 IAM API 删除 AWSServiceRoleForElasticBeanstalk 服务相关角色。有关更多信息，请参阅《IAM 用户指南》的删除服务相关角色。

Elastic Beanstalk 服务相关角色支持的区域

Elastic Beanstalk 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS Elastic Beanstalk 终端节点和配额。

维护服务相关角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与 Elastic Beanstalk 直接关联的独特类型的 IAM 角色。服务相关角色由 Elastic Beanstalk 预定义，包括该服务代表您调用其他服务所需的所有权限。 AWS

您可以使用服务相关角色轻松设置 Elastic Beanstalk，因为您不必手动添加所需的权限。Elastic Beanstalk 定义了其服务相关角色的权限，除非另外定义，否则只有 Elastic Beanstalk 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 Elastic Beanstalk 资源，因为您无法无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的AWS 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。选择 Yes（是）和链接，查看该服务的服务相关角色文档。

Elastic Beanstalk 的服务相关角色权限

Elastic Beanstalk 使用名为的服务相关角色 AWSServiceRoleForElasticBeanstalkMaintenance——允许 Elastic Beanstalk 为您的运行环境执行定期维护活动。

AWSServiceRoleForElasticBeanstalkMaintenance 服务相关角色信任以下服务来代入该角色：

- `maintenance.elasticbeanstalk.amazonaws.com`

AWSServiceRoleForElasticBeanstalkMaintenance 服务相关角色的权限策略包含 Elastic Beanstalk 代表您完成操作所需的所有权限：

```
{
    "Version": "2012-10-17",
    "Statement":
        {
            "Sid": "AllowCloudformationChangeSetOperationsOnElasticBeanstalkStacks",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:ListChangeSets",
                "cloudformation:DescribeStacks"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/awseb-*",
                "arn:aws:cloudformation:*:*:stack/eb-*"
            ]
        }
```

```
}
```

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

或者，您可以使用 AWS 托管策略提供对 Elastic Beanstalk 的完全访问权限。

为 Elastic Beanstalk 创建服务相关角色

您无需手动创建服务相关角色。当您使用 Elastic Beanstalk API 创建 Elastic Beanstalk 环境但未指定实例配置文件时，Elastic Beanstalk 会为您创建服务相关角色。

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。如果您在 2019 年 4 月 18 日之前使用 Elastic Beanstalk 服务，当时该服务开始 AWSServiceRoleForElasticBeanstalkMaintenance 支持服务相关角色，而您的账户需要该服务，那么 Elastic Beanstalk 会在您的账户中创建该角色。AWSServiceRoleForElasticBeanstalkMaintenance 要了解更多信息，请参阅我的 IAM 账户中的新角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您使用 Elastic Beanstalk API 创建 Elastic Beanstalk 环境但未指定实例配置文件时，Elastic Beanstalk 会再次为您创建服务相关角色。

为 Elastic Beanstalk 编辑服务相关角色

Elastic Beanstalk 不允许你编辑服务相关角色。 AWSServiceRoleForElasticBeanstalkMaintenance 创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参见《IAM 用户指南》中的编辑服务相关角色。

删除 Elastic Beanstalk 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

清除服务相关角色

在使用 IAM 删除服务相关角色之前，必须先终止使用此服务相关角色的任何 Elastic Beanstalk 环境。

> **Note**
>
> 在您尝试终止环境时，如果 Elastic Beanstalk 服务在使用服务相关角色，终止操作可能会失败。如果发生这种情况，请等待几分钟后重试。

终止使用（控制台）的 Elastic Beanstalk 环境 AWSServiceRoleForElasticBeanstalkMaintenance

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

有关使用 EB CLI 终止 Elastic Beanstalk 环境的详细信息，请参阅eb terminate。

有关TerminateEnvironment使用 API 终止 Elastic Beanstalk 环境的详细信息，请参阅。

手动删除服务相关角色

使用 IAM 控制台、IAM CLI 或 IAM API 删除 AWSServiceRoleForElasticBeanstalkMaintenance 服务相关角色。有关更多信息，请参阅《IAM 用户指南》的删除服务相关角色。

Elastic Beanstalk 服务相关角色支持的区域

Elastic Beanstalk 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS Elastic Beanstalk 终端节点和配额。

托管更新服务相关角色

AWS Elastic Beanstalk 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与 Elastic Beanstalk 直接关联的独特类型的 IAM 角色。服务相关角色由 Elastic Beanstalk 预定义，包括该服务代表您调用其他服务所需的所有权限。 AWS

您可以使用服务相关角色轻松设置 Elastic Beanstalk，因为您不必手动添加所需的权限。Elastic Beanstalk 定义了其服务相关角色的权限，除非另外定义，否则只有 Elastic Beanstalk 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 Elastic Beanstalk 资源，因为您无法无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的AWS 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。选择 Yes（是）和链接，查看该服务的服务相关角色文档。

Elastic Beanstalk 的服务相关角色权限

Elastic Beanstalk 使用名为的服务相关角色 AWSServiceRoleForElasticBeanstalkManagedUpdates——允许 Elastic Beanstalk 对正在运行的环境执行定期平台更新。

AWSServiceRoleForElasticBeanstalkManagedUpdates 服务相关角色信任以下服务来代入该角色：

- `managedupdates.elasticbeanstalk.amazonaws.com`

托管策略AWSElasticBeanstalkManagedUpdatesServiceRolePolicy允许 AWSServiceRoleForElasticBeanstalkManagedUpdates 服务相关角色获得 Elastic Beanstalk 代表您完成托管更新操作所需的所有权限。要查看托管策略的内容，请参阅《AWS 托管策略参考指南》中的AWSElasticBeanstalkManagedUpdatesServiceRolePolicy页面。

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

或者，您可以使用 AWS 托管策略提供对 Elastic Beanstalk 的完全访问权限。

为 Elastic Beanstalk 创建服务相关角色

您无需手动创建服务相关角色。当您使用 Elastic Beanstalk API 创建 Elastic Beanstalk 环境、启用托管更新并指定 `AWSServiceRoleForElasticBeanstalkManagedUpdates` 作为 `aws:elasticbeanstalk:managedactions` 命名空间 `ServiceRoleForManagedUpdates` 选项的值时，Elastic Beanstalk 会为您创建服务相关角色。

当 Elastic Beanstalk 在 AWSServiceRoleForElasticBeanstalkManagedUpdates 创建环境时尝试为您的账户创建服务相关角色时，您必须拥有该权限。`iam:CreateServiceLinkedRole`如果您没有此权限，环境创建将失败，您会看到说明问题的消息。

作为替代方案，也可以让具有创建服务相关角色权限的另一用户使用 IAM 预先创建服务相关角色。这样，即使您没有 `iam:CreateServiceLinkedRole` 权限，也可以创建环境。

您（或其他用户）可以使用 IAM 控制台，通过 Elastic Beanstalk 托管更新使用案例创建服务相关角色。在 IAM CLI 或 IAM API 中，用 `managedupdates.elasticbeanstalk.amazonaws.com` 服务名称创建一个服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您使用 Elastic Beanstalk API 创建 Elastic Beanstalk 环境、启用托管更新并指定 `AWSServiceRoleForElasticBeanstalkManagedUpdates` 作为 `aws:elasticbeanstalk:managedactions` 命名空间 `ServiceRoleForManagedUpdates` 选项的值时，Elastic Beanstalk 会再次为您创建服务相关角色。

为 Elastic Beanstalk 编辑服务相关角色

Elastic Beanstalk 不允许你编辑服务相关角色。AWSServiceRoleForElasticBeanstalkManagedUpdates 创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参见《IAM 用户指南》中的编辑服务相关角色。

删除 Elastic Beanstalk 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

清除服务相关角色

您必须首先确保启用了托管更新的 Elastic Beanstalk 环境使用的是不同服务角色或已终止，然后才能使用 IAM 删除服务相关角色。

> **Note**
>
> 在您尝试终止环境时，如果 Elastic Beanstalk 服务在使用服务相关角色，终止操作可能会失败。如果发生这种情况，请等待几分钟后重试。

终止使用（控制台）的 Elastic Beanstalk 环境 AWSServiceRoleForElasticBeanstalkManagedUpdates

1. 打开 Elastic Beanstalk 控制台，然后在 “区域” 列表中，选择您的。 AWS 区域

2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Terminate Environment（终止环境）。
4. 使用屏幕上的对话框确认环境终止。

有关使用 EB CLI 终止 Elastic Beanstalk 环境的详细信息，请参阅eb terminate。

有关TerminateEnvironment使用 API 终止 Elastic Beanstalk 环境的详细信息，请参阅。

手动删除服务相关角色

使用 IAM 控制台、IAM CLI 或 IAM API 删除 AWSServiceRoleForElasticBeanstalkManagedUpdates 服务相关角色。有关更多信息，请参阅《IAM 用户指南》的删除服务相关角色。

Elastic Beanstalk 服务相关角色支持的区域

Elastic Beanstalk 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS Elastic Beanstalk 终端节点和配额。

管理 Elastic Beanstalk 用户策略

AWS Elastic Beanstalk 提供了两个托管策略，允许您为 Elastic Beanstalk 管理的所有资源分配完全访问权限或只读访问权限。您可以将策略附加到 AWS Identity and Access Management (IAM) 用户或群组，或者关联到您的用户担任的角色。

托管用户策略

- AdministratorAccess-AWSElasticBeanstalk — 向用户授予创建、修改和删除 Elastic Beanstalk 应用程序、应用程序版本、配置设置、环境及其底层资源的完全管理权限。要查看托管策略的内容，请参阅《AWS 托管策略参考指南》中的 AdministratorAccess-AWSElasticBeanstalk 页面。
- AWSElasticBeanstalkReadOnly— 允许用户查看应用程序和环境，但不允许执行修改应用程序和环境的操作。它提供对所有 Elastic Beanstalk 资源以及 Elastic Beanstalk 控制台检索 AWS 的其他资源的只读访问权限。请注意，只读访问不会启用下载 Elastic Beanstalk 日志等操作，以便您阅读它们。这是因为日志存储在 Amazon S3 存储桶中，Elastic Beanstalk 需要其写入权限。有关如何启用对 Elastic Beanstalk 日志的访问，请参阅本主题结尾的示例。要查看托管策略的内容，请参阅《AWS 托管策略参考指南》中的AWSElasticBeanstalkReadOnly页面。

> **Important**
>
> Elastic Beanstalk 托管式策略不提供精细权限—它们授予使用 Elastic Beanstalk 应用程序可能需要的所有权限。在某些情况下，您可能希望进一步限制我们的托管策略的权限。有关一个用例的示例，请参阅防止跨环境访问 Amazon S3 存储桶。
> 我们的托管式策略也不涵盖对您可能添加到解决方案中且不由 Elastic Beanstalk 管理的自定义资源的权限。要实施更精细的权限、所需的最低权限或自定义资源权限，请使用自定义策略。

已弃用的 托管策略

以前，Elastic Beanstalk 支持另外两个托管用户策略，以及。AWSElasticBeanstalkFullAccessAWSElasticBeanstalkReadOnlyAccess我们计划停用以前的这些策略。您可能仍然能够在 IAM 控制台中查看和使用它们。但是，我们建议您切换到使用新的托管用户策略，然后添加自定义策略以授予对自定义资源的权限（如果有的话）。

与其他服务集成的策略

如果您更喜欢使用其他服务，则会提供更精细的策略，以允许您将环境与其他服务集成。

- AWSElasticBeanstalkRoleCWL— 允许环境管理 Amazon CloudWatch 日志组。
- AWSElasticBeanstalkRoleRDS— 允许环境集成 Amazon RDS 实例。
- AWSElasticBeanstalkRoleWorkerTier— 允许工作环境层创建亚马逊 DynamoDB 表和亚马逊 SQS 队列。
- AWSElasticBeanstalkRoleECS— 允许多容器 Docker 环境管理亚马逊 ECS 集群。
- AWSElasticBeanstalkRoleCore— 允许 Web 服务环境的核心操作。
- AWSElasticBeanstalkRoleSNS— 允许环境启用 Amazon SNS 主题集成。

要查看特定托管策略的 JSON 来源，请参阅AWS 托管策略参考指南。

使用托管式策略控制访问

您可以使用托管式策略授予对 Elastic Beanstalk 的完全访问权限或只读访问权限。Elastic Beanstalk 在需要额外权限来访问新功能时自动更新这些策略。

将托管式策略应用于 IAM 用户或组

1. 在 IAM 控制台中打开 Policies（策略）页。

2. 在搜索框中键入 **AWSElasticBeanstalk** 以筛选策略。
3. 在策略列表中，选中AWSElasticBeanstalkReadOnly或 AdministratorAccess- 旁边的复选框 AWSElasticBeanstalk。
4. 选择 Policy actions（策略操作），然后选择 Attach（附加）。
5. 选择一个或多个要将策略附加到的用户和组。您可以使用 Filter（筛选条件）菜单和搜索框来筛选委托人实体列表。
6. 选择附加策略。

创建自定义用户策略

您可以创建自己的 IAM policy，以允许或拒绝针对特定 Elastic Beanstalk 资源执行特定的 Elastic Beanstalk API 操作，并控制对不由 Elastic Beanstalk 管理的自定义资源的访问权限。有关将策略附加到用户或组的详细信息，请参阅《IAM 用户指南》中的使用策略。有关创建自定义策略的详细信息，请参阅《IAM 用户指南》中的创建 IAM 策略。

> **Note**
>
> 虽然您可以限制用户与 Elastic Beanstalk API 交互的方式，但当前没有有效的方式来阻止有权创建必需基础资源的用户在 Amazon EC2 和其他服务中创建其他资源。
> 将这些策略视为分发 Elastic Beanstalk 责任的有效方式，而不是视为保护所有基础资源的方式。

2019 年 11 月，Elastic Beanstalk 发布了对 Amazon EC2 启动模板的支持。这是一种新的资源类型，可供环境 Auto Scaling 组用于启动 Amazon EC2 实例，并且该资源类型需要新的权限。如果您的用户策略缺乏所需权限，环境仍可以使用传统资源启动配置。因此大多数客户都不会受到影响。但是，如果您尝试使用需要 Amazon EC2 启动模板的新功能，并且您具有自定义策略，您的环境创建或更新可能会失败。在这种情况下，请确保您的自定义策略具有以下权限。

Amazon EC2 启动模板所需的权限

- `EC2:CreateLaunchTemplate`
- `EC2:CreateLaunchTemplateVersions`
- `EC2:DeleteLaunchTemplate`
- `EC2:DeleteLaunchTemplateVersions`
- `EC2:DescribeLaunchTemplate`

- `EC2:DescribeLaunchTemplateVersions`

IAM policy 包含策略语句，这些语句描述了您要授予的权限。为 Elastic Beanstalk 创建策略语句时，您需要了解如何使用策略语句的以下四个部分：

- Effect（效果）指定是允许还是拒绝该语句中的操作。
- Action（操作）指定您要控制的 API 操作。例如，使用 `elasticbeanstalk:CreateEnvironment` 指定 `CreateEnvironment` 操作。某些操作（如创建环境）需要额外的权限才能执行。有关更多信息，请参阅Elastic Beanstalk 操作的资源和条件。

> **Note**
>
> 要使用 `UpdateTagsForResource` API 操作，请指定以下两个虚拟操作之一（或两者），而不是 API 操作名称：
>
> `elasticbeanstalk:AddTags`
>
> 控制调用 `UpdateTagsForResource` 和传递要在 `TagsToAdd` 参数中添加的标签列表的权限。
>
> `elasticbeanstalk:RemoveTags`
>
> 控制调用 `UpdateTagsForResource` 和传递要在 `TagsToRemove` 参数中删除的标签键列表的权限。

- Resource（资源）指定您要控制访问权限的资源。要指定 Elastic Beanstalk 资源，请列出各个资源的 Amazon Resource Name（ARN）。
- （可选）条件指定对语句中授予的权限的限制。有关更多信息，请参阅 Elastic Beanstalk 操作的资源和条件。

以下部分说明了几种可能需要考虑自定义用户策略的情况。

启用有限的 Elastic Beanstalk 环境创建

以下示例中的策略可让用户调用 `CreateEnvironment` 操作，从而使用指定应用程序和应用程序版本创建名称以 **`Test`** 开头的环境。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid":"CreateEnvironmentPerm",
      "Action": [
        "elasticbeanstalk:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My First Elastic
 Beanstalk Application/Test*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My First Elastic Beanstalk Application"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:applicationversion/My First Elastic Beanstalk Application/First
 Release"]
        }
      }
    },
    {
      "Sid":"AllNonResourceCalls",
      "Action":[
        "elasticbeanstalk:CheckDNSAvailability",
        "elasticbeanstalk:CreateStorageLocation"
      ],
      "Effect":"Allow",
      "Resource":[
        "*"
      ]
    }
  ]
}
```

以上策略显示如何授予对 Elastic Beanstalk 操作的有限访问权限。为了实际启动环境，用户还必须拥有创建为环境提供支持的 AWS 资源的权限。例如，以下策略授予对 Web 服务器环境的默认资源集的访问权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "ec2:*",
        "ecs:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "sqs:*"
        ],
      "Resource": "*"
    }
  ]
}
```

启用对存储在 Amazon S3 中的 Elastic Beanstalk 日志的访问

以下示例中的策略可让用户提取 Elastic Beanstalk 日志、在 Amazon S3 中暂存日志和检索日志。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:DeleteObject",
        "s3:GetObjectAcl",
        "s3:PutObjectAcl"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::elasticbeanstalk-*"
    }
  ]
}
```

> **Note**
>
> 要将这些权限限制为仅日志路径，请使用以下资源格式。
>
> ```
> "arn:aws:s3:::elasticbeanstalk-us-east-2-123456789012/resources/environments/
> logs/*"
> ```

启用特定 Elastic Beanstalk 应用程序的管理

以下示例中的策略可让用户管理一个特定的 Elastic Beanstalk 应用程序中的环境和其他资源。此策略拒绝 Elastic Beanstalk 对其他应用程序的资源执行操作，并拒绝创建和删除 Elastic Beanstalk 应用程序。

> **Note**
>
> 该策略不拒绝通过其他服务来访问任何资源。它展示的是在各个用户之间分发 Elastic Beanstalk 应用程序的管理责任的有效方式，而不是保护基础资源的方式。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:CreateApplication",
        "elasticbeanstalk:DeleteApplication"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:CreateApplicationVersion",
        "elasticbeanstalk:CreateConfigurationTemplate",
        "elasticbeanstalk:CreateEnvironment",
        "elasticbeanstalk:DeleteApplicationVersion",
        "elasticbeanstalk:DeleteConfigurationTemplate",
        "elasticbeanstalk:DeleteEnvironmentConfiguration",
        "elasticbeanstalk:DescribeApplicationVersions",
        "elasticbeanstalk:DescribeConfigurationOptions",
        "elasticbeanstalk:DescribeConfigurationSettings",
        "elasticbeanstalk:DescribeEnvironmentResources",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticbeanstalk:DescribeEvents",
        "elasticbeanstalk:DeleteEnvironmentConfiguration",
        "elasticbeanstalk:RebuildEnvironment",
```

```
        "elasticbeanstalk:RequestEnvironmentInfo",
        "elasticbeanstalk:RestartAppServer",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "elasticbeanstalk:SwapEnvironmentCNAMEs",
        "elasticbeanstalk:TerminateEnvironment",
        "elasticbeanstalk:UpdateApplicationVersion",
        "elasticbeanstalk:UpdateConfigurationTemplate",
        "elasticbeanstalk:UpdateEnvironment",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "elasticbeanstalk:ValidateConfigurationSettings"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringNotEquals": {
          "elasticbeanstalk:InApplication": [
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/myapplication"
          ]
        }
      }
    }
  ]
}
```

Elastic Beanstalk 的 Amazon 资源名称格式

使用该资源的 Amazon Resource Name（ARN）为 IAM 策略指定资源。对于 Elastic Beanstalk，ARN 的格式如下。

```
arn:aws:elasticbeanstalk:region:account-id:resource-type/resource-path
```

其中：

- *region* 是资源所在的区域（例如，**us-west-2**）。
- *account-id* 是 AWS 账户 ID，不包含连字符（例如，**123456789012**）。
- *resource-type* 标识 Elastic Beanstalk 资源的类型，例如 。`environment`有关所有 Elastic Beanstalk 资源类型的列表，请参阅下表。
- *resource-path* 是用于标识特定资源的部分。Elastic Beanstalk 资源具有唯一标识该资源的路径。请参阅下表，查看每种资源类型的资源路径格式。例如，环境与应用程序是始终关联在一起的。在应用程序 **myEnvironment** 中，环境 **myApp** 的资源路径将如下所示：

```
myApp/myEnvironment
```

Elastic Beanstalk 有多种类型的资源，可供您在策略中指定使用。下表显示的是各个资源类型的 ARN 格式及示例。

资源类型	ARN 格式
application	arn:aws:elasticbeanstalk: *region*:*account-id* :application/ *application-name* 示例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App**
applicationversion	arn:aws:elasticbeanstalk: *region*:*account-id* :applicationversion/ *application-name* /*version-label* 示例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version**
configurationtemplate	arn:aws:elasticbeanstalk: *region*:*account-id* :configurationtemplate/ *application-name* /*template-name* 示例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template**
environment	arn:aws:elasticbeanstalk: *region*:*account-id* :environment/ *application-name* /*environment-name* 示例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/MyEnvironment**
platform	arn:aws:elasticbeanstalk: *region*:*account-id* :platform/ *platform-name* /*platform-version* 示例：**arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/MyPlatform/1.0**

资源类型	ARN 格式
solutionstack	arn:aws:elasticbeanstalk: *region*::solutionstack/ *solutionstack-name* 示例：**arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7**

环境、应用程序版本和配置模板始终包含在特定的应用程序内。您将会注意到，这些资源在它们的资源路径中都有一个应用程序名，以便通过它们的资源名和包含的应用程序对它们进行唯一标识。虽然解决方案堆栈是供配置模板和环境使用的，但是它们并非是针对某一应用程序或 AWS 账户的，且它们的 ARN 中不存在应用程序或 AWS 账户。

Elastic Beanstalk 操作的资源和条件

本部分描述了可在策略语句中用于授予权限的资源和条件，这些权限允许用户对特定的 Elastic Beanstalk 资源执行特定的 Elastic Beanstalk 操作。

条件可让您指定完成此操作所需的资源的权限。例如，当您调用 CreateEnvironment 操作时，还必须指定要部署的应用程序版本及包含此应用程序名称的应用程序。为 CreateEnvironment 操作设置权限时，应使用 InApplication 和 FromApplicationVersion 条件指定您要执行操作的应用程序和应用程序版本。

此外，还可以使用解决方案堆栈（FromSolutionStack）或配置模板（FromConfigurationTemplate）指定环境配置。以下策略语句允许 CreateEnvironment 操作，借助 **myenv** 配置（Resource）使用应用程序版本 **My App**（InApplication）在应用程序 **My Version**（由 FromApplicationVersion 条件指定）中创建名为 **32bit Amazon Linux running Tomcat 7**（由 FromSolutionStack 指定）的环境：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:CreateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:applicationversion/My App/My Version"],
          "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-
east-2::solutionstack/32bit Amazon Linux running Tomcat 7"]
        }
      }
    }
  ]
}
```

> **Note**
>
> 本主题中提及的大多数条件键特定于 Elastic Beanstalk，而且其名称包含 `elasticbeanstalk:` 前缀。为简洁起见，我们会在以下部分中提交条件键名称时从这些名称中忽略此前缀。例如，我们会提及 `InApplication` 而不是其全名 `elasticbeanstalk:InApplication`。
> 相反，我们会提及跨 AWS 服务所使用的一些条件键，而且我们包含其 `aws:` 前缀以突出显示异常。
> 策略示例始终显示完整条件键名称，包括前缀。

小节目录

Elastic Beanstalk 操作的策略信息

下表列出了所有 Elastic Beanstalk 操作、每项操作针对的资源以及可以使用条件提供的其他上下文信息。

Elastic Beanstalk 操作的策略信息，包括资源、条件、示例和依赖项

资源	条件	示例语句
操作：AbortEnvironmentUpdate		

application
 	
environment
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许用户在名为 My App 的应用程序中中止有关环境的环境更新操作。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:AbortEnvironmentUpdate"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"
		]
		}
		]
		}

操作：CheckDNSAvailability		

"*"	不适用	{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:CheckDNSAvailability"
		],
		"Effect": "Allow",
		"Resource": "*"
		}
		]
		}

资源	条件	示例语句
操作：ComposeEnvironments		

application	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许用户编写属于名为 My App 的应用程序的环境。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:ComposeEnvironments"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App"
		]
		}
		]
		}

操作：CreateApplication		

资源	条件	示例语句
application	aws:RequestTag/ *key-name*（可选） aws:TagKeys （可选）	此示例允许 CreateApplication 操作创建名称以 **DivA** 开头的应用程序： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:CreateApplication" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/DivA*" ] } ] }`
操作：CreateApplicationVersion		

资源	条件	示例语句

applicationversion	InApplication
	
	aws:RequestTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		此示例允许 CreateApplicationVersion 操作在应用程序 * 中创建使用任一名称（**My App**）的应用程序版本：
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:CreateApplicationVersion"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/*"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

操作：CreateConfigurationTemplate

资源	条件	示例语句
configurationtemplate	InApplication FromApplication FromApplicationVersion FromConfigurationTemplate FromEnvironment FromSolutionStack aws:RequestTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 CreateConfigurationTemplate 操作在应用程序 **My Template** 中创建名称以 My Template* （**My App**）开头的配置模板： `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:CreateConfigurationTemplate"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template*"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"], "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7"] } } }] }`
操作：CreateEnvironment		

资源	条件	示例语句
environment	InApplication FromApplicationVersion FromConfigurationTemplate FromSolutionStack aws:RequestTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 CreateEnvironment 操作在应用程序 **myenv** 中使用解决方案堆栈 **My App** 创建名为 **32bit Amazon Linux running Tomcat 7** 的环境： `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:CreateEnvironment"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"], "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"], "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-east-2::solutionstack/32bit Amazon Linux running Tomcat 7"] } } }] }`
操作：CreatePlatformVersion		

资源	条件	示例语句
platform	aws:RequestTag/ *key-name*（可选） aws:TagKeys （可选）	此示例允许 CreatePlatformVersion 操作创建以 us-east-2 区域为目标的平台版本，其名称以 **us-east-2_** 开头： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:CreatePlatformVersion" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*" ] } ] }`
操作：CreateStorageLocation		
"*"	不适用	`{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:CreateStorageLocation" ], "Effect": "Allow", "Resource": "*" } ] }`
操作：DeleteApplication		

资源	条件	示例语句

application	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 DeleteApplication 操作删除应用程序 **My App**：
		
		`{`
		`  "Version": "2012-10-17",`
		`  "Statement": [`
		`  {`
		`  "Action": [`
		`  "elasticbeanstalk:DeleteApplication"`
		`  ],`
		`  "Effect": "Allow",`
		`  "Resource": [`
		`  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"`
		`  ]`
		`  }`
		`  ]`
		`}`

操作：DeleteApplicationVersion		

资源	条件	示例语句
applicationversion	InApplication aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DeleteApplicationVersion 操作在应用程序 **My Version** 中删除名为 **My App** 的应用程序版本： `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:DeleteApplicationVersion"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
操作：DeleteConfigurationTemplate		

资源	条件	示例语句
configurationtemplate	InApplication （可选） aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DeleteConfigurationTemplate 操作在应用程序 **My Template** 中删除名为 **My App** 的配置模板。将应用程序名称指定为条件（可选）。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DeleteConfigurationTemplate" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template" ] } ] }`
操作：DeleteEnvironmentConfiguration		

资源	条件	示例语句

environment	InApplication （可选）	以下策略允许 DeleteEnvironmentConfiguration 操作在应用程序 **myenv** 中删除环境 **My App** 的预配置。将应用程序名称指定为条件（可选）。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:DeleteEnvironmentConfiguration"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		]
		}
		]
		}

操作：DeletePlatformVersion		

资源	条件	示例语句
platform	aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DeletePlatformVersion 操作删除以 us-east-2 区域为目标的平台版本，其名称以 **us-east-2_** 开头： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DeletePlatformVersion" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*" ] } ] }`
操作：DescribeApplications		

资源	条件	示例语句

application	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 DescribeApplications 操作描述应用程序“My App”。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:DescribeApplications"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"
		]
		}
		]
		}

操作：DescribeApplicationVersions		

资源	条件	示例语句
applicationversion	InApplication （可选） aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DescribeApplicationVersions 操作在应用程序 **My Version** 中描述应用程序版本 **My App**。将应用程序名称指定为条件（可选）。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DescribeApplicationVersions" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version" ] } ] }`
操作：DescribeConfigurationOptions		

资源	条件	示例语句
environment configurationtemplate solutionstack	InApplication （可选） aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DescribeConfigurationOptions 操作在应用程序 **myenv** 中描述环境 **My App** 的配置选项。将应用程序名称指定为条件（可选）。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeConfigurationOptions", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
操作：DescribeConfigurationSettings		

资源	条件	示例语句

environment
 	
configurationtemplate
	InApplication （可选）
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 DescribeConfigurationSettings 操作在应用程序 **myenv** 中描述环境 **My App** 的配置设置。将应用程序名称指定为条件（可选）。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": "elasticbeanstalk:DescribeConfigurationSettings",
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		]
		}
		]
		}

操作：DescribeEnvironmentHealth		

资源	条件	示例语句
environment	aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许使用 DescribeEnvironmentHealth 检索名为 **myenv** 的环境的运行状况信息。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeEnvironmentHealth", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
操作：DescribeEnvironmentResources		

资源	条件	示例语句
environment	InApplication （可选） aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DescribeEnvironmentResources 操作在应用程序 **My App** 中返回环境 **myenv** 的 AWS 资源列表。将应用程序名称指定为条件（可选）。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeEnvironmentResources", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
操作：DescribeEnvironments		

资源	条件	示例语句
environment	InApplication （可选） aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DescribeEnvironments 操作在应用程序 **myenv** 中描述环境 **myotherenv** 和 **My App**。将应用程序名称指定为条件（可选）。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeEnvironments", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv", "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App2/myotherenv" ] } ] }`
操作：DescribeEvents		

资源	条件	示例语句
application applicationversion configurationtemplate environment	InApplication aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DescribeEvents 操作在应用程序 **myenv** 中列出环境 **My Version** 和应用程序版本 **My App** 的事件描述。 `{ "Version": "2012-10-17", "Statement": [{ "Action": "elasticbeanstalk:DescribeEvents", "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv", "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
操作：DescribeInstancesHealth		

资源	条件	示例语句
environment	不适用	以下策略允许使用 DescribeInstancesHealth 检索名为 **myenv** 的环境中的实例的运行状况信息。 `{ "Version": "2012-10-17", "Statement": [ { "Action": "elasticbeanstalk:DescribeInstancesHealth", "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv" ] } ] }`
操作：DescribePlatformVersion		

资源	条件	示例语句
platform	aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 DescribePlatformVersion 操作描述以 us-east-2 区域为目标的平台版本，其名称以 **us-east-2_** 开头： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:DescribePlatformVersion" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*" ] } ] }`
操作：ListAvailableSolutionStacks		

资源	条件	示例语句

solutions tack	不适用	以下策略允许 ListAvailableSolutionStacks 操作仅返回解决方案堆栈 **32bit Amazon Linux running Tomcat 7**。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:ListAvail
		ableSolutionStacks"
		],
		"Effect": "Allow",
		"Resource": "arn:aws:elasticbe
		anstalk:us-east-2::solutionstack/32bit
		Amazon Linux running Tomcat 7"
		}
		]
		}

操作：ListPlatformVersions

资源	条件	示例语句
platform	aws:RequestTag/ *key-name*（可选） aws:TagKeys （可选）	此示例允许 CreatePlatformVersion 操作创建以 us-east-2 区域为目标的平台版本，其名称以 **us-east-2_** 开头： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:ListPlatformVersions" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:platform/us-east-2_*" ] } ] }`
操作：ListTagsForResource		

资源	条件	示例语句

application
 	
applicationversion
 	
configurationtemplate
 	
environment
 	
platform
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略仅在现有资源具有名为 ListTagsForResource 的带有值 stage 的标签时允许 test 操作列出现有资源的标签。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:ListTagsForResource"
		],
		"Effect": "Allow",
		"Resource": "*",
		"Condition": {
		"StringEquals": {
		"aws:ResourceTag/stage": ["test"]
		}
		}
		}
		]
		}

操作：RebuildEnvironment		

资源	条件	示例语句

environment	InApplication
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 RebuildEnvironment 操作在应用程序 **myenv** 中重建环境 **My App**。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:RebuildEnvironment"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

操作：RequestEnvironmentInfo		

资源	条件	示例语句
environment	InApplication aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 RequestEnvironmentInfo 操作在应用程序 **myenv** 中编译有关环境 **My App** 的信息。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:RequestEnvironmentInfo"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
操作：RestartAppServer		

资源	条件	示例语句

environment	InApplication	以下策略允许 RestartAppServer 操作在应用程序 **myenv** 中重启环境 **My App** 的应用程序容器服务器。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:RestartAppServer"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

操作：RetrieveEnvironmentInfo		

资源	条件	示例语句
environment	InApplication aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 RetrieveEnvironmentInfo 操作在应用程序 **myenv** 中检索环境 **My App** 的已编译信息。 `{ "Version": "2012-10-17", "Statement": [{ "Action": ["elasticbeanstalk:RetrieveEnvironmentInfo"], "Effect": "Allow", "Resource": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"], "Condition": { "StringEquals": { "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"] } } }] }`
操作：SwapEnvironmentCNAMEs		

资源	条件	示例语句

environment	InApplication （可选）
	
	FromEnvironment （可选）
		以下策略允许 SwapEnvironmentCNAMEs 操作交换环境 **mysrcenv** 和 **mydestenv** 的别名记录。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:SwapEnvironmentCNAMEs"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/mysrcenv",
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/mydestenv"
		]
		}
		]
		}

操作：TerminateEnvironment		

资源	条件	示例语句

environment	InApplication
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 TerminateEnvironment 操作在应用程序 **myenv** 中终止环境 **My App**。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:TerminateEnvironment"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

操作：UpdateApplication		

资源	条件	示例语句
application	aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 UpdateApplication 操作更新应用程序 **My App** 的属性。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:UpdateApplication" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App" ] } ] }`
操作：UpdateApplicationResourceLifecycle		

资源	条件	示例语句
application	aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	以下策略允许 UpdateApplicationResourceLifecycle 操作更新应用程序 **My App** 的生命周期设置。 `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:UpdateApplicationResourceLifecycle" ], "Effect": "Allow", "Resource": [ "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App" ] } ] }`
操作：UpdateApplicationVersion		

资源	条件	示例语句

applicationversion	InApplication
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 UpdateApplicationVersion 操作在应用程序 **My Version** 中更新应用程序版本 **My App** 的属性。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:UpdateApplicationVersion"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

操作：UpdateConfigurationTemplate		

资源	条件	示例语句

configurationtemplate	InApplication
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 UpdateConfigurationTemplate 操作在应用程序 **My Template** 中更新配置模板 **My App** 的属性或选项。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:UpdateConfigurationTemplate"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My Template"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

操作：UpdateEnvironment		

资源	条件	示例语句

environment	InApplication
	
	FromApplicationVersion
	
	FromConfigurationTemplate
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 UpdateEnvironment 操作通过部署应用程序版本 **myenv** 在应用程序 **My App** 中更新环境 **My Version**。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:UpdateEnvironment"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"],
		"elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/My Version"]
		}
		}
		}
		]
		}

操作：UpdateTagsForResource – AddTags		

资源	条件	示例语句

application
 	
applicationversion
 	
configurationtemplate
 	
environment
 	
platform
	aws:ResourceTag/ *key-name*（可选）
	
	aws:RequestTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		AddTags 操作是与 UpdateTagsForResource API 关联的两个虚拟操作之一。
		
		以下策略仅在现有资源具有名为 AddTags 的带有值 stage 的标签时允许 test 操作修改现有资源的标签。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:AddTags"
		],
		"Effect": "Allow",
		"Resource": "*",
		"Condition": {
		"StringEquals": {
		"aws:ResourceTag/stage": ["test"]
		}
		}
		}
		]
		}

操作：UpdateTagsForResource – RemoveTags		

资源	条件	示例语句
application applicationversion configurationtemplate environment platform	aws:ResourceTag/ *key-name*（可选） aws:TagKeys （可选）	RemoveTags 操作是与 UpdateTagsForResource API 关联的两个虚拟操作之一。 以下策略拒绝请求从现有资源中删除名为 RemoveTags 的标签的 stage 操作： `{ "Version": "2012-10-17", "Statement": [ { "Action": [ "elasticbeanstalk:RemoveTags" ], "Effect": "Deny", "Resource": "*", "Condition": { "ForAnyValue:StringEquals": { "aws:TagKeys": ["stage"] } } } ] }`
操作：ValidateConfigurationSettings		

资源	条件	示例语句

template
 	
environment
	InApplication
	
	aws:ResourceTag/ *key-name*（可选）
	
	aws:TagKeys （可选）
		以下策略允许 ValidateConfigurationSettings 操作在应用程序 **myenv** 中根据环境 **My App** 验证配置设置。
		
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Action": [
		"elasticbeanstalk:ValidateConfigurationSettings"
		],
		"Effect": "Allow",
		"Resource": [
		"arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
		],
		"Condition": {
		"StringEquals": {
		"elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-east-2:123456789012:application/My App"]
		}
		}
		}
		]
		}

Elastic Beanstalk 操作的条件键

密钥可让您指定用于表达依赖项、限制权限的条件，或指定某一操作的输入参数约束。Elastic Beanstalk 支持以下键。

`InApplication`

指定相关应用程序，其中包含了供操作运行的资源。

以下示例允许 UpdateApplicationVersion 操作更新应用程序版本 **My Version** 的属性。InApplication 条件将 **My App** 指定为 **My Version** 的容器。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateApplicationVersion"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/My App/
My Version"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"]
        }
      }
    }
  ]
}
```

FromApplicationVersion

将应用程序版本指定为输入参数的依赖项或约束。

以下示例允许 UpdateEnvironment 操作在应用程序 **myenv** 中更新环境 **My App**。FromApplicationVersion 条件会限制 VersionLabel 参数，仅允许应用程序版本 **My Version** 更新此环境。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
```

```
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromApplicationVersion": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:applicationversion/My App/My Version"]
        }
      }
    }
  ]
}
```

FromConfigurationTemplate

将配置模板指定为输入参数的依赖项或约束。

以下示例允许 UpdateEnvironment 操作在应用程序 **myenv** 中更新环境 **My App**。FromConfigurationTemplate 条件会限制 TemplateName 参数，仅允许配置模板 **My Template** 更新此环境。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:UpdateEnvironment"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/myenv"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromConfigurationTemplate":
 ["arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My App/My
 Template"]
        }
      }
    }
```

```
  ]
}
```

FromEnvironment

将环境指定为输入参数的依赖项或约束。

以下示例允许 SwapEnvironmentCNAMEs 操作在 **My App** 中的名称以 **mysrcenv** 和 **mydestenv** 开头的所有环境之间交换别名记录，但这不适用于名称以 **mysrcenvPROD*** 和 **mydestenvPROD*** 开头的环境。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:SwapEnvironmentCNAMEs"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mysrcenv*",
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mydestenv*"
      ],
      "Condition": {
        "StringNotLike": {
          "elasticbeanstalk:FromEnvironment": [
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mysrcenvPROD*",
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/My App/
mydestenvPROD*"
          ]
        }
      }
    }
  ]
}
```

FromSolutionStack

将解决方案堆栈指定为输入参数的依赖项或约束。

以下策略允许 CreateConfigurationTemplate 操作在应用程序 **My Template** 中创建名称以 My Template*（**My App**）开头的配置模板。FromSolutionStack 条件会限制 solutionstack 参数，仅允许将解决方案堆栈 **32bit Amazon Linux running Tomcat 7** 用作该参数的输入值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticbeanstalk:CreateConfigurationTemplate"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticbeanstalk:us-east-2:123456789012:configurationtemplate/My
 App/My Template*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticbeanstalk:InApplication": ["arn:aws:elasticbeanstalk:us-
east-2:123456789012:application/My App"],
          "elasticbeanstalk:FromSolutionStack": ["arn:aws:elasticbeanstalk:us-
east-2::solutionstack/32bit Amazon Linux running Tomcat 7"]
        }
      }
    }
  ]
}
```

aws:ResourceTag/*key-name*, aws:RequestTag/*key-name*, aws:TagKeys

指定基于标签的条件。有关详细信息，请参阅使用标签控制对 Elastic Beanstalk 资源的访问。

使用标签控制对 Elastic Beanstalk 资源的访问

AWS Identity and Access Management（IAM）用户策略语句中的条件是您用于指定 Elastic Beanstalk 操作需要完成的资源的权限的语法的一部分。有关指定策略语句条件的详细信息，请参阅Elastic Beanstalk 操作的资源和条件。使用条件中的标签是控制对资源和请求的访问的一种方法。有关标记 Elastic Beanstalk 资源的信息，请参阅标记 Elastic Beanstalk 应用程序资源。本主题讨论了基于标签的访问控制。

在设计 IAM 策略时，您可以通过授予对特定资源的访问权限来设置精细权限。但随着您管理的资源数量的增加，此任务会变得日益复杂。标记资源并在策略声明条件中使用标签可以简化这一任务。您可以向具有特定标签的任何资源批量授予访问权限。然后，在创建期间或之后，您可以将此标签反复应用到相关资源。

标签可以附加到资源，也可以从请求传入支持标签的服务。在 Elastic Beanstalk 中，资源可以具有标签，而且某些操作可以包括标签。在创建 IAM 策略时，您可以使用标签条件键来控制：

- 哪些用户可以基于环境已有的标签对环境执行操作。
- 哪些标签可以在操作的请求中传递。
- 是否特定标签键可在请求中使用。

有关标签条件键的完整请求和语义，请参阅《IAM 用户指南》中的使用标签控制访问。

以下示例演示如何为 Elastic Beanstalk 用户指定策略中的标签条件。

Example 1：基于请求中的标签限制操作

Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 托管式用户策略为用户提供对任意 Elastic Beanstalk 托管式资源执行任意 Elastic Beanstalk 操作的无限权限。

以下策略限制此权力并拒绝未经授权的用户创建 Elastic Beanstalk 生产环境的权限。为此，如果请求指定一个名为 `CreateEnvironment` 的带有值为 `stage` 或 `gamma` 的标签，则它会拒绝 `prod` 操作。此外，该策略还通过不允许标签修改操作来阻止这些未经授权的用户篡改生产环境的阶段以包含这些的标签值或完全删除 `stage` 标签。除托管用户策略外，客户的管理员还必须将此 IAM 策略附加到未经授权的 IAM 用户。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:CreateEnvironment",
        "elasticbeanstalk:AddTags"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/stage": ["gamma", "prod"]
```

```
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:RemoveTags"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": ["stage"]
        }
      }
    }
  ]
}
```

Example 2：基于资源标签限制操作

Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 托管式用户策略为用户提供对任意 Elastic Beanstalk 托管式资源执行任意 Elastic Beanstalk 操作的无限权限。

以下策略限制此权力并拒绝未经授权的用户对 Elastic Beanstalk 生产环境执行操作的权限。为此，如果环境具有名为 `stage` 的带有值 `gamma` 或 `prod` 的标签，则它会拒绝特定操作。除托管用户策略外，客户的管理员还必须将此 IAM 策略附加到未经授权的 IAM 用户。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticbeanstalk:AddTags",
        "elasticbeanstalk:RemoveTags",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticbeanstalk:TerminateEnvironment",
        "elasticbeanstalk:UpdateEnvironment",
        "elasticbeanstalk:ListTagsForResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/stage": ["gamma", "prod"]
        }
      }
    }
  ]
}
```

Example 3：基于请求中的标签允许操作

以下策略授予用户创建 Elastic Beanstalk 开发应用程序的权限。

为此，如果请求指定一个名为 `stage` 的带有值为 `development` 的标签，则它允许 `CreateApplication` 和 `AddTags` 操作。`aws:TagKeys` 条件可确保用户无法添加其他标签键。特别是，它能确保 `stage` 标签键的大小写正确。请注意，此策略对于未附加 Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 托管式用户策略的 IAM 用户非常有用。此托管策略为用户提供对任意 Elastic Beanstalk 托管资源执行任意 Elastic Beanstalk 操作的无限权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticbeanstalk:CreateApplication",
        "elasticbeanstalk:AddTags"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/stage": "development"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["stage"]
        }
      }
    }
  ]
}
```

Example 4：基于资源标签允许操作

以下策略授予用户对 Elastic Beanstalk 开发应用程序执行操作以及获取其相关信息的权限。

为此，如果应用程序具有名为 stage、值为 development 的标签，则它允许执行特定操作。aws:TagKeys 条件可确保用户无法添加其他标签键。特别是，它能确保 stage 标签键的大小写正确。请注意，此策略对于未附加 Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 托管式用户策略的 IAM 用户非常有用。此托管策略为用户提供对任意 Elastic Beanstalk 托管资源执行任意 Elastic Beanstalk 操作的无限权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticbeanstalk:UpdateApplication",
        "elasticbeanstalk:DeleteApplication",
        "elasticbeanstalk:DescribeApplications"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/stage": "development"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["stage"]
        }
      }
    }
  ]
}
```

基于托管式策略的示例策略

本节演示了如何控制用户对 AWS Elastic Beanstalk 的访问，并包含了为常见场景提供所需访问权限的示例策略。这些策略派生自 Elastic Beanstalk 托管式策略。有关将托管式策略附加到用户和组的信息，请参见管理 Elastic Beanstalk 用户策略。

在此情景中，Example Corp. 是一家软件公司，有三个团队负责公司网站：即管理基础设施的管理员团队、构建网站软件的开发人员团队及测试网站的 QA 团队。为帮助管理他们的 Elastic Beanstalk 资源权限，Example Corp. 公司创建了每个相应团队成员所属的三个组：管理员组、开发人员组和测试人员组。Example Corp. 公司可为管理员组提供针对所有应用程序、环境及其底层资源的完全访问权限，以便他们创建、诊断和删除所有 Elastic Beanstalk 资产。开发人员需要获取查看所有 Elastic Beanstalk 资产以及创建并部署应用程序版本的权限。开发人员不能创建新应用程序或环境，也不能终止正在运行

的环境。测试人员需要查看所有 Elastic Beanstalk 资源来监控和测试应用程序。测试人员不应具有更改任何 Elastic Beanstalk 资源的权限。

以下示例策略为每个组提供了所需权限。

示例 1：管理员组 – 所有 Elastic Beanstalk 和相关服务 API

以下策略向用户授予使用 Elastic Beanstalk 时必需的所有操作权限。此策略还允许 Elastic Beanstalk 代表您在以下服务中配置和管理资源。在创建环境时，Elastic Beanstalk 依靠这些附加服务来配置底层资源。

- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- Auto Scaling
- Amazon CloudWatch
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon Relational Database Service
- AWS CloudFormation

请注意，此策略是一个示例。它为 AWS 服务提供了宽泛的权限，Elastic Beanstalk 可使用这些权限管理应用程序和环境。例如，`ec2:*` 允许 AWS Identity and Access Management（IAM）用户对 AWS 账户中的任何 Amazon EC2 资源执行任何操作。这些权限并不限于与 Elastic Beanstalk 配合使用的资源。作为最佳实践，您仅应向个人授予他们履行职责所需的权限。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
```

```
        "rds:*",
        "cloudformation:*"
      ],
      "Resource" : "*"
    }
  ]
}
```

示例 2：开发人员组 – 除高权限以外的所有人员

以下策略拒绝授予创建应用程序和环境的权限，但允许执行所有其他的 Elastic Beanstalk 操作。

请注意，此策略是一个示例。它为 AWS 产品提供了宽泛的权限，Elastic Beanstalk 可使用这些权限管理应用程序和环境。例如，`ec2:*` 允许 IAM 用户对 AWS 账户中的任何 Amazon EC2 资源执行任何操作。这些权限并不限于与 Elastic Beanstalk 配合使用的资源。作为最佳实践，您仅应向个人授予他们履行职责所需的权限。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Action" : [
        "elasticbeanstalk:CreateApplication",
        "elasticbeanstalk:CreateEnvironment",
        "elasticbeanstalk:DeleteApplication",
        "elasticbeanstalk:RebuildEnvironment",
        "elasticbeanstalk:SwapEnvironmentCNAMEs",
        "elasticbeanstalk:TerminateEnvironment"],
      "Effect" : "Deny",
      "Resource" : "*"
    },
    {
      "Action" : [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "rds:*",
        "cloudformation:*"],
      "Effect" : "Allow",
```

```
      "Resource" : "*"
    }
  ]
}
```

示例 3：测试人员 – 仅限查看

以下策略允许给所有应用程序、应用程序版本、事件和环境提供只读访问权限。它不允许执行任何操作。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticbeanstalk:Check*",
        "elasticbeanstalk:Describe*",
        "elasticbeanstalk:List*",
        "elasticbeanstalk:RequestEnvironmentInfo",
        "elasticbeanstalk:RetrieveEnvironmentInfo",
        "ec2:Describe*",
        "elasticloadbalancing:Describe*",
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:List*",
        "cloudwatch:Get*",
        "s3:Get*",
        "s3:List*",
        "sns:Get*",
        "sns:List*",
        "rds:Describe*",
        "cloudformation:Describe*",
        "cloudformation:Get*",
        "cloudformation:List*",
        "cloudformation:Validate*",
        "cloudformation:Estimate*"
      ],
      "Resource" : "*"
    }
  ]
}
```

基于资源权限的示例策略

本部分介绍了一个使用案例，用于说明如何控制 Elastic Beanstalk 操作（访问特定 Elastic Beanstalk 资源）的用户权限。我们将介绍支持此使用案例的示例策略。有关 Elastic Beanstalk 资源的更多信息，请参阅创建自定义用户策略。有关将策略附加到用户和组的信息，请转到《使用 AWS Identity and Access Management》中的管理 IAM 策略。

在使用案例中，Example Corp. 是一家为两类不同客户开发应用程序的小型咨询公司。John 是开发经理，负责监管 app1 和 app2 这两种 Elastic Beanstalk 应用程序的开发。John 会对这两种应用程序执行一些开发和测试工作，且只有他能为这两种应用程序更新生产环境。对于 app1 和 app2，他需要拥有以下权限：

- 查看应用程序、应用程序版本、环境和配置模板
- 创建应用程序版本并将它们部署到过渡环境
- 更新生产环境
- 创建和终止环境

Jill 是一名测试人员，为监控和测试这两种应用程序，她需要拥有以下资源的查看权限：应用程序、应用程序版本、环境和配置模板。但是，她不应具有更改任何 Elastic Beanstalk 资源的权限。

Jack 是 app1 的开发人员，需要拥有查看所有 app1 资源的权限，且还需要为 app1 创建应用程序版本并将应用程序版本部署到过渡环境。

Judy 是 Example Corp AWS 账户的管理员。她已为 John、Jill 和 Jack 创建了 IAM 用户并将以下策略附加到这些用户，从而针对 app1 和 app2 应用程序授予相应权限。

示例 1：John – app1、app2 的开发经理

我们已将 John 的策略细分成三项独立策略，以便易于读取和管理它们。通过结合这些示例，可授予 John 对这两个应用程序执行开发、测试和部署操作所需的权限。

第一个策略指定了 Auto Scaling、Amazon S3、Amazon EC2、CloudWatch、Amazon SNS、Elastic Load Balancing、Amazon RDS 和 AWS CloudFormation 的操作。在创建环境时，Elastic Beanstalk 依靠这些附加服务来配置底层资源。

请注意，此策略是一个示例。它为 AWS 产品提供了宽泛的权限，Elastic Beanstalk 可使用这些权限管理应用程序和环境。例如，`ec2:*` 允许 IAM 用户对 AWS 账户中的任何 Amazon EC2 资源执行任何操作。这些权限并不限于与 Elastic Beanstalk 配合使用的资源。作为最佳实践，您仅应向个人授予他们履行职责所需的权限。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "ecs:*",
            "ecr:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "cloudformation:*",
            "dynamodb:*",
            "rds:*",
            "sqs:*",
            "logs:*",
            "iam:GetPolicyVersion",
            "iam:GetRole",
            "iam:PassRole",
            "iam:ListRolePolicies",
            "iam:ListAttachedRolePolicies",
            "iam:ListInstanceProfiles",
            "iam:ListRoles",
            "iam:ListServerCertificates",
            "acm:DescribeCertificate",
            "acm:ListCertificates",
            "codebuild:CreateProject",
            "codebuild:DeleteProject",
            "codebuild:BatchGetBuilds",
            "codebuild:StartBuild"
         ],
         "Resource":"*"
      }
   ]
}
```

第二项策略指定了 John 可以对 app1 和 app2 资源执行的 Elastic Beanstalk 操作。`AllCallsInApplications` 语句允许对 app1 和 app2 内的所有资源执行任何 Elastic Beanstalk 操作（`"elasticbeanstalk:*"`）（例

如，elasticbeanstalk:CreateEnvironment）。AllCallsOnApplications 语句允许对 app1 和 app2 应用程序资源执行任何 Elastic Beanstalk 操作（"elasticbeanstalk:*"）（例如，elasticbeanstalk:DescribeApplications、elasticbeanstalk:UpdateApplication 等）。AllCallsOnSolutionStacks 语句允许对解决方案堆栈资源执行任何 Elastic Beanstalk 操作（"elasticbeanstalk:*"）（例如，elasticbeanstalk:ListAvailableSolutionStacks）。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllCallsInApplications",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
               ]
            }
         }
      },
      {
         "Sid":"AllCallsOnApplications",
         "Action":[
            "elasticbeanstalk:*"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
         ]
      },
      {
         "Sid":"AllCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:*"
```

```
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2::solutionstack/*"
         ]
      }
   ]
}
```

第三项策略指定了第二项策略需要获取 Elastic Beanstalk 操作权限才能完成的那些 Elastic Beanstalk 操作。AllNonResourceCalls 语句允许执行 elasticbeanstalk:CheckDNSAvailability 操作（即调用 elasticbeanstalk:CreateEnvironment 所需的操作）及其他操作。此语句还允许执行 elasticbeanstalk:CreateStorageLocation 操作（即 elasticbeanstalk:CreateApplication、elasticbeanstalk:CreateEnvironment 所需的操作）及其他操作。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

示例 2：Jill – app1、app2 的测试人员

我们已将 Jill 的策略细分成三项独立策略，以便易于读取和管理它们。通过结合这些示例，可授予 Jill 对这两种应用程序执行测试和监控操作所需的权限。

第一个策略指定了针对 Auto Scaling、Amazon S3、Amazon EC2、CloudWatch、Amazon SNS、Elastic Load Balancing、Amazon RDS 和 AWS CloudFormation 的 Describe*、List* 和

Get* 操作（适用于非早期容器类型），使得 Elastic Beanstalk 操作可以检索有关 app1 和 app2 应用程序的底层资源的信息。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "ec2:Describe*",
            "elasticloadbalancing:Describe*",
            "autoscaling:Describe*",
            "cloudwatch:Describe*",
            "cloudwatch:List*",
            "cloudwatch:Get*",
            "s3:Get*",
            "s3:List*",
            "sns:Get*",
            "sns:List*",
            "rds:Describe*",
            "cloudformation:Describe*",
         "cloudformation:Get*",
         "cloudformation:List*",
         "cloudformation:Validate*",
         "cloudformation:Estimate*"
         ],
         "Resource":"*"
      }
   ]
}
```

第二项策略指定了 Jill 可以对 app1 和 app2 资源执行的 Elastic Beanstalk 操作。AllReadCallsInApplications 语句允许 Jill 调用 Describe* 操作和环境信息操作。AllReadCallsOnApplications 语句允许 Jill 对 app1 和 app2 应用程序资源调用 DescribeApplications 和 DescribeEvents 操作。AllReadCallsOnSolutionStacks 语句允许对解决方案堆栈资源的执行查看操作（ListAvailableSolutionStacks、DescribeConfigurationOptions 和 ValidateConfigurationSettings）。

```
{
   "Version": "2012-10-17",
   "Statement":[
```

```
      {
         "Sid":"AllReadCallsInApplications",
         "Action":[
            "elasticbeanstalk:Describe*",
            "elasticbeanstalk:RequestEnvironmentInfo",
            "elasticbeanstalk:RetrieveEnvironmentInfo"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnApplications",
         "Action":[
            "elasticbeanstalk:DescribeApplications",
            "elasticbeanstalk:DescribeEvents"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1",
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app2"
         ]
      },
      {
         "Sid":"AllReadCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:ListAvailableSolutionStacks",
            "elasticbeanstalk:DescribeConfigurationOptions",
            "elasticbeanstalk:ValidateConfigurationSettings"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2::solutionstack/*"
         ]
      }
```

```
   ]
}
```

第三项策略指定了第二项策略需要获取 Elastic Beanstalk 操作权限才能完成的那些 Elastic Beanstalk 操作。`AllNonResourceCalls` 语句允许执行 `elasticbeanstalk:CheckDNSAvailability` 操作，这是一些查看操作所需要的操作。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

示例 3：Jack – app1 的开发人员

我们已将 Jack 的策略细分成三项独立策略，以便易于读取和管理它们。通过结合这些示例，可授予 Jack 对 app1 资源执行测试、监控和部署操作所需的权限。

第一个策略指定了针对 Auto Scaling、Amazon S3、Amazon EC2、CloudWatch、Amazon SNS、Elastic Load Balancing、Amazon RDS 和 AWS CloudFormation 的操作（适用于非早期容器类型），使得 Elastic Beanstalk 操作可以处理 app1 的底层资源。有关支持的非早期容器类型的列表，请参阅the section called “为什么某些平台版本标记为传统版本？”

请注意，此策略是一个示例。它为 AWS 产品提供了宽泛的权限，Elastic Beanstalk 可使用这些权限管理应用程序和环境。例如，`ec2:*` 允许 IAM 用户对 AWS 账户中的任何 Amazon EC2 资源执行任何操作。这些权限并不限于与 Elastic Beanstalk 配合使用的资源。作为最佳实践，您仅应向个人授予他们履行职责所需的权限。

```
{
   "Version": "2012-10-17",
   "Statement":[
```

```
      {
         "Effect":"Allow",
         "Action":[
            "ec2:*",
            "elasticloadbalancing:*",
            "autoscaling:*",
            "cloudwatch:*",
            "s3:*",
            "sns:*",
            "rds:*",
            "cloudformation:*"
         ],
         "Resource":"*"
      }
   ]
}
```

第二项策略指定了 Jack 可以对 app1 资源执行的 Elastic Beanstalk 操作。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllReadCallsAndAllVersionCallsInApplications",
         "Action":[
            "elasticbeanstalk:Describe*",
            "elasticbeanstalk:RequestEnvironmentInfo",
            "elasticbeanstalk:RetrieveEnvironmentInfo",
            "elasticbeanstalk:CreateApplicationVersion",
            "elasticbeanstalk:DeleteApplicationVersion",
            "elasticbeanstalk:UpdateApplicationVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1"
               ]
            }
         }
```

```
      },
      {
         "Sid":"AllReadCallsOnApplications",
         "Action":[
            "elasticbeanstalk:DescribeApplications",
            "elasticbeanstalk:DescribeEvents"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1"
         ]
      },
      {
         "Sid":"UpdateEnvironmentInApplications",
         "Action":[
            "elasticbeanstalk:UpdateEnvironment"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2:123456789012:environment/app1/app1-
staging*"
         ],
         "Condition":{
            "StringEquals":{
               "elasticbeanstalk:InApplication":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:application/app1"
               ]
            },
            "StringLike":{
               "elasticbeanstalk:FromApplicationVersion":[
                  "arn:aws:elasticbeanstalk:us-east-2:123456789012:applicationversion/
app1/*"
               ]
            }
         }
      },
      {
         "Sid":"AllReadCallsOnSolutionStacks",
         "Action":[
            "elasticbeanstalk:ListAvailableSolutionStacks",
            "elasticbeanstalk:DescribeConfigurationOptions",
            "elasticbeanstalk:ValidateConfigurationSettings"
         ],
         "Effect":"Allow",
```

```
         "Resource":[
            "arn:aws:elasticbeanstalk:us-east-2::solutionstack/*"
         ]
      }
   ]
}
```

第三项策略指定了第二项策略需要获取 Elastic Beanstalk 操作权限才能完成的那些 Elastic Beanstalk 操作。`AllNonResourceCalls` 语句允许执行 `elasticbeanstalk:CheckDNSAvailability` 操作（即调用 `elasticbeanstalk:CreateEnvironment` 所需的操作）及其他操作。此语句还允许执行 `elasticbeanstalk:CreateStorageLocation` 操作（即 `elasticbeanstalk:CreateEnvironment` 所需的操作）及其他操作。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllNonResourceCalls",
         "Action":[
            "elasticbeanstalk:CheckDNSAvailability",
            "elasticbeanstalk:CreateStorageLocation"
         ],
         "Effect":"Allow",
         "Resource":[
            "*"
         ]
      }
   ]
}
```

防止跨环境访问 Amazon S3 存储桶

Elastic Beanstalk 提供托管策略来 AWS 处理您账户中 Elastic Beanstalk 环境所需的资源。 AWS 默认情况下，向您的 AWS 账户中的一个应用程序提供的权限可以访问属于同一 AWS 账户中其他应用程序的 S3 资源。

如果您的 AWS 账户运行多个 Beanstalk 应用程序，则可以创建自己的自定义策略，将其附加到您自己的服务角色或每个环境的实例配置文件，从而缩小策略的安全范围。然后，您可以将自定义策略中的 S3 权限限制为特定环境。

> **Note**
>
> 请注意，您有责任维护您的自定义政策。如果您的自定义策略所依据的 Elastic Beanstalk 托管策略发生变化，则您需要根据对基本策略的相应更改来修改您的自定义策略。有关 Elastic Beanstalk 托管策略的变更历史记录，请参阅。Elastic Bean AWS stalk 更新了托管策略

缩小权限范围的示例

以下示例基于AWSElasticBeanstalkWebTier托管策略。

默认策略包括以下几行，用于对 S3 存储桶的权限。此默认策略不将 S3 存储桶操作限制在特定的环境或应用程序中。

```
{
   "Sid" : "BucketAccess",
   "Action" : [
      "s3:Get*",
      "s3:List*",
      "s3:PutObject"
     ],
   "Effect" : "Allow",
   "Resource" : [
      "arn:aws:s3:::elasticbeanstalk-*",
      "arn:aws:s3:::elasticbeanstalk-*/*"
     ]
}
```

您可以通过将特定资源限定为的服务角色来缩小访问范围`Principal`。以下示例提供对环境中 S3 存储桶的自定义服务角色`aws-elasticbeanstalk-ec2-role-my-example-env`权限，ID `my-example-env-ID` 为。

Example 仅向特定环境的 S3 存储桶授予权限

```
{
   "Sid": "BucketAccess",
   "Action": [
      "s3:Get*",
      "s3:List*",
      "s3:PutObject"
```

```
    ],
   "Effect": "Allow",
   "Principal": {
      "AWS": "arn:aws:iam::...:role/aws-elasticbeanstalk-ec2-role-my-example-env"
     },
   "Resource": [
      "arn:aws:s3:::elasticbeanstalk-my-region-account-id-12345",
      "arn:aws:s3:::elasticbeanstalk-my-region-account-id-12345/resources/environments/
my-example-env-ID/*"
    ]
}
```

> **Note**
>
> 资源 ARN 必须包含 Elastic Beanstalk 环境 ID（不是环境名称）。您可以从 Elastic Beanstalk 控制台的环境概述页面上获取环境 ID。您也可以使用 desc AWS CLI ribe-environments 命令来获取此信息。

有关帮助您更新 Elastic Beanstalk 环境的 S3 存储桶权限的更多信息，请参阅以下资源：

- 本指南中的将 Elastic Beanstalk 和 Amazon S3 结合使用
- Amazon S3 在服务授权参考指南中定义的资源类型
- IA M 用户指南中的 ARN 格式

将 Elastic Beanstalk 和 Amazon RDS 结合使用

您可以将 Elastic Beanstalk 与 Amazon Relational Database Service（Amazon RDS）结合使用，来设置、操作和扩展关系数据库。分别有如下两种选项可以开始使用。

- 在 Amazon RDS 中创建新的数据库
- 从先前由 Elastic Beanstalk 创建并随后与 Beanstalk 环境解耦的数据库开始。有关更多信息，请参阅the section called “数据库”。

您可以使用任一方法在 Amazon RDS 中运行数据库实例，并将应用程序配置为在启动时连接到该数据库实例。您可以将多个环境连接到数据库，还可以通过蓝绿部署执行无缝更新。

> **Note**
>
> 如果您之前没有将数据库实例与应用程序结合使用，我们建议您请先使用 Elastic Beanstalk 控制台将一个数据库实例添加到测试环境。这样，您就可以验证应用程序是否可以读取环境属性、构建连接字符串以及连接到数据库实例，而无需为独立的数据库执行额外的配置工作。有关更多信息，请参阅将数据库添加到 Elastic Beanstalk 环境。

要允许环境中的 Amazon EC2 实例连接到外部数据库，请为与环境关联的 Auto Scaling 组配置一个额外安全组。您可以将同一安全组附加到数据库实例。或者，您可以使用单独的安全组。如果您附加了其他安全组，则必须将附加到数据库的安全组配置为允许来自此安全组的入站访问。

> **Note**
>
> 您可以通过向附加到数据库的安全组添加规则来将环境连接到数据库。此规则必须允许从 Elastic Beanstalk 附加到您的环境的 Auto Scaling 组的自动生成的安全组进行入站访问。但是，要知道，通过创建此规则，您还可以在两个安全组之间建立依赖关系。随后，当您尝试终止环境时，Elastic Beanstalk 将无法删除环境的安全组，因为数据库的安全组依赖于环境的安全组。

在启动数据库实例并配置安全组后，您可以使用环境属性将连接信息（例如端点和密码）传递到应用程序。这是当您在环境中运行数据库实例时，Elastic Beanstalk 在后台使用的相同机制。

为了提供一层额外的安全性，您可以将连接信息存储在 Amazon S3 中，并将 Elastic Beanstalk 配置为在部署期间检索该信息。利用配置文件（`.ebextensions`），您可以在部署应用程序时配置环境中的实例以从 Amazon S3 安全地检索文件。

主题

在默认 VPC 中启动并连接到外部 Amazon RDS 实例

要将外部数据库与在 Elastic Beanstalk 中运行的应用程序结合使用，您有两种选项可以选择。或者，您可以使用 Amazon RDS 启动数据库实例。使用 Amazon RDS 启动的任何实例均完全独立于 Elastic Beanstalk 和您的 Elastic Beanstalk 环境。这意味着，您可以使用 Amazon RDS 支持的任何数据库引擎和实例类型，甚至是 Elastic Beanstalk 未使用的数据库引擎和实例类型。

或者，作为启动新数据库实例的替代方案，您可以从先前由 Elastic Beanstalk 创建并随后与 Beanstalk 环境解耦的数据库开始。有关更多信息，请参阅the section called “数据库”。使用此选项后，您无需完成启动新数据库的过程。但是，您确实需要完成本主题中介绍的后续过程。

以下过程介绍默认 VPC 的流程。如果您使用的是自定义 VPC，该流程也是相同的。唯一的附加要求是环境和数据库实例位于同一子网中，或位于可以互相通信的子网中。有关配置与 Elastic Beanstalk 一起使用的自定义 VPC 的更多信息，请参阅将 Elastic Beanstalk 和 Amazon VPC 结合使用。

> **Note**
>
> - 如果您从 Elastic Beanstalk 创建并随后与 Beanstalk 环境解耦的数据库开始，则可以跳过第一组步骤，然后按 To modify the inbound rules on your RDS instance's security group（修改 RDS 实例的安全组上的入站规则）下分组的步骤继续。
> - 如果您计划为生产环境使用解耦数据库，请验证数据库使用的存储类型适合您的工作负载。有关更多信息，请参阅《Amazon RDS 用户指南》中的数据库实例存储和修改数据库实例。

在默认 VPC 中启动 RDS 数据库实例

1. 打开 RDS 控制台。
2. 在导航窗格中，选择 Databases（数据库）。
3. 选择创建数据库。
4. 选择 Standard Create（标准创建）。

   > **Important**
   >
   > 请勿选择 Easy Create（轻松创建）。如果您选择它，您将无法配置启动此 RDS 数据库所需的设置。

5. 在 Additional configuration（附加配置）下，对于 Initial database name（初始数据库名称），键入 **ebdb**。

6. 请查看默认设置并根据您的具体要求调整这些设置。请注意以下选项：

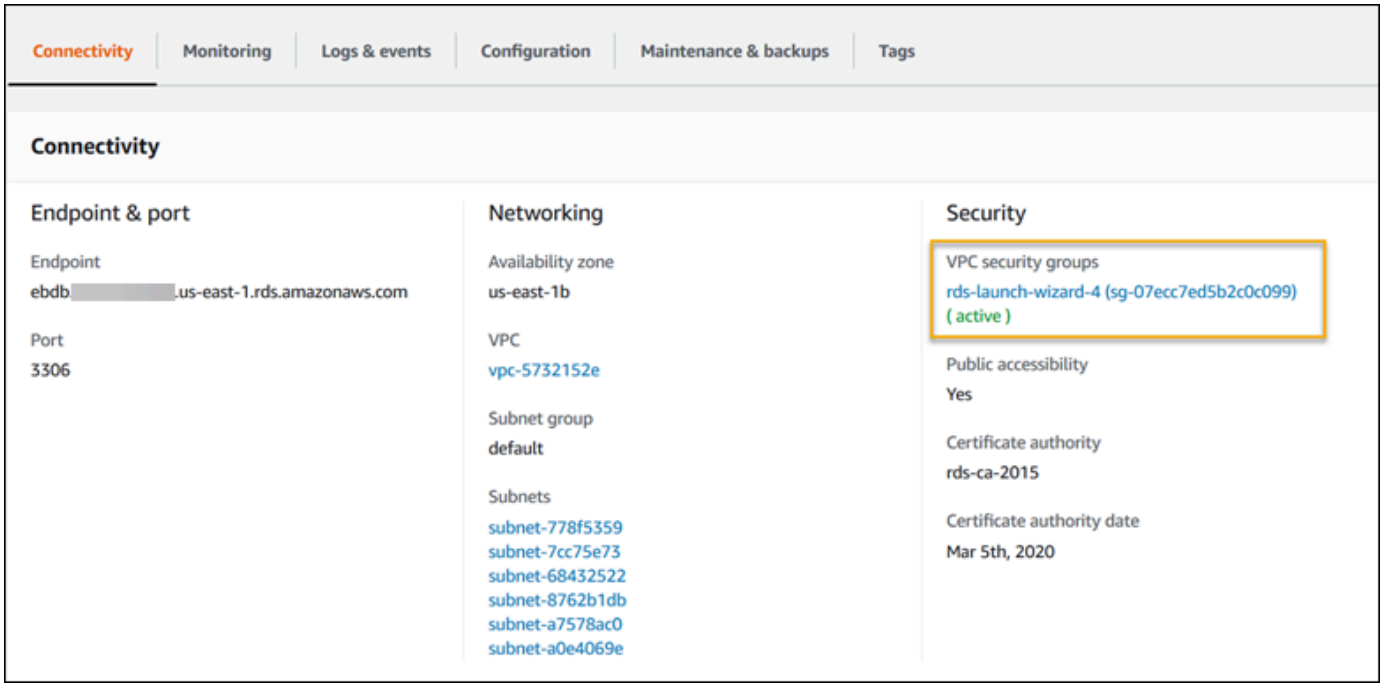
   - DB instance class（数据库实例类）– 选择对于您的工作负载具有适当的内存量和 CPU 能力的实例大小。
   - Multi-AZ deployment（多可用区部署）– 为了实现高可用性，请将其设置为 Create an Aurora Replica/Reader node in a different AZ（在不同的可用区中创建 Aurora 副本/读取器节点）。
   - Master username（主用户名）和 Master password（主密码）– 数据库用户名和密码。请记下这些设置，因为您以后将使用这些值。

7. 验证其余选项的默认设置，然后选择 Create database（创建数据库）。

接下来，修改附加到数据库实例的安全组以允许相应的端口上的入站流量。这与您稍后要附加到 Elastic Beanstalk 环境的安全组相同。因此，您添加的规则将向同一安全组中的其他资源授予入站访问权限。

修改附加到 RDS 实例的安全组上的入站规则

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例的名称以查看其详细信息。
4. 在 Connectivity（连接）部分中，记下在该页上显示的 Subnets（子网）、Security groups（安全组）和 Endpoint（端点）。这样您稍后可以使用这些信息。
5. 在 Security（安全性）下面，您可以查看与数据库实例关联的安全组。打开链接以在 Amazon EC2 控制台中查看安全组。



6. 在安全组详细信息中，选择 Inbound（入站）选项卡。
7. 选择 Edit（编辑）。
8. 选择 Add Rule（添加规则）。
9. 对于 Type（类型），请选择应用程序使用的数据库引擎。
10. 对于 Source（源），键入 **sg-** 以查看可用安全组的列表。请选择 Elastic Beanstalk 环境中使用的与 Auto Scaling 组关联的安全组。这样，环境中的 Amazon EC2 实例就可以访问数据库。

11. 选择 Save（保存）。

接下来，将数据库实例的安全组添加到您的运行环境。在此过程中，Elastic Beanstalk 使用附加的其他安全组重新预置环境中的所有实例。

向环境添加安全组

- 请执行下列操作之一：
  - 使用 Elastic Beanstalk 控制台添加安全组

    a. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。

    b. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

    > **Note**
    >
    > 如果您有多个环境，请使用搜索栏筛选环境列表。

    c. 在导航窗格中，选择 Configuration（配置）。

    d. 在 Instances（实例）配置类别中，选择 Edit（编辑）。

    e. 在 EC2 security groups（EC2 安全组）下面，选择要附加到实例的安全组以及 Elastic Beanstalk 创建的实例安全组。

    f. 要保存更改，请选择页面底部的 Apply（应用）。

    g. 阅读警告，然后选择 Confirm（确认）。
  - 要使用配置文件添加安全组，请使用 `securitygroup-addexisting.config` 示例文件。

接下来，使用环境属性将连接信息传递到环境。在使用 Elastic Beanstalk 控制台向环境中添加数据库实例时，Elastic Beanstalk 使用 RDS_HOSTNAME 等环境属性将连接信息传递到您的应用程序。您可以使用相同的属性。这样，您就可以为集成的数据库实例和外部的数据库实例使用相同的应用程序代码。或者，您可以选择自己的属性名称。

为 Amazon RDS 数据库实例配置环境属性

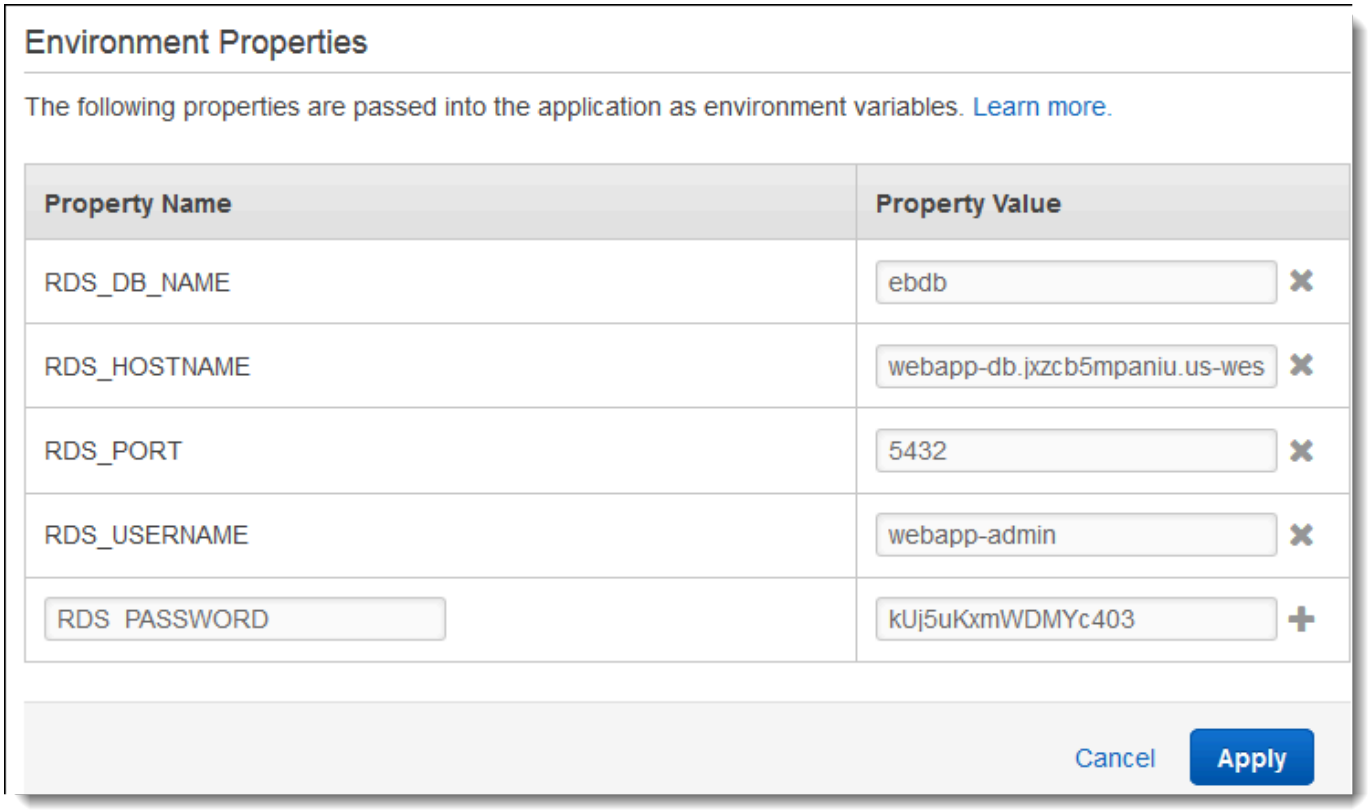
1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

   > **Note**
   >
   > 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。

5. 在环境属性部分中，定义应用程序读取的用于构建连接字符串的变量。为了实现与具有集成 RDS 数据库实例的环境的兼容，请使用以下名称和值。您可以在 RDS 控制台中找到除密码以外的所有值。

属性名称	描述	属性值
RDS_HOSTNAME	数据库实例的主机名。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Endpoint（端点）。
RDS_PORT	数据库实例接受连接的端口。默认值因数据库引擎而异。	在 Amazon RDS 控制台的 Connectivity & security（连接和安全）选项卡上：Port（端口）。
RDS_DB_NAME	数据库名称 **ebdb**。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：DB Name（数据库名称）。
RDS_USERNAME	您为数据库配置的用户名。	在 Amazon RDS 控制台的 Configuration（配置）选项卡上：Master username（主用户名）。
RDS_PASSWORD	您为数据库配置的密码。	在 Amazon RDS 控制台中不可供参考。



6. 要保存更改，请选择页面底部的 Apply（应用）。

如果您尚未将应用程序编程为读取环境属性和构建连接字符串，请参阅以下特定于语言的主题以获得指导：

- Java SE – 连接数据库 (Java SE 平台)
- Java with Tomcat – 连接数据库 (Tomcat 平台)
- Node.js – 连接到数据库
- .NET – 连接到数据库
- PHP – 使用 PDO 或 MySQLi 连接到数据库
- Python – 连接到数据库
- Ruby – 连接到数据库

最后，根据应用程序读取环境变量的时间，您可能需要在环境中的实例上重新启动应用程序服务器。

重新启动您的环境的应用程序服务器

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Restart app server(s)（重启应用程序服务器）。

在 EC2 Classic 中启动并连接到外部 Amazon RDS 实例

> **Important**
>
> Amazon EC2-Classic 将于 2022 年 8 月 15 日之后结束标准支持。为避免工作负载中断，我们建议您在相应日期之前从 Amazon EC2-Classic 迁移到 VPC。我们还要求您以后不要启动任何 Amazon EC2-Classic 上的 AWS 资源，而是使用 Amazon VPC 代替。有关更多信息，请参阅从 EC2-Classic 迁移到 VPC和 EC2-Classic Networking is Retiring - Here's How to Prepare 博客文章。

如果您将 EC2 Classic（无 VPC）与 AWS Elastic Beanstalk 结合使用，该过程会由于安全组的工作方式的差异而略微改变。在 EC2 Classic 中，数据库实例无法使用 EC2 安全组，因此会获取只适用于 Amazon RDS 的数据库安全组。

您可以向数据库安全组添加规则以允许从 EC2 安全组进行入站访问。但是，您不能将数据库安全组附加到与环境关联的 Auto Scaling 组。要避免在数据库安全组和您的环境之间建立依赖关系，您必须在 Amazon EC2 中创建第三个安全组。然后，您需要在数据库安全组中添加规则以授予对新安全组的入站访问权限。最后，您应该将其分配给您的 Elastic Beanstalk 环境中的 Auto Scaling 组。

> **Note**
>
> - 如果您从 Elastic Beanstalk 创建并随后与 Beanstalk 环境解耦的数据库开始，则可以跳过第一组步骤，然后按 To create a bridge security group（创建桥接安全组）下分组的步骤继续。

- 如果您计划为生产环境使用解耦数据库，请验证数据库使用的存储类型适合您的工作负载。有关更多信息，请参阅《Amazon RDS 用户指南》中的数据库实例存储和修改数据库实例。

在 EC2 classic（无 VPC）中启动 RDS 实例

1. 打开 RDS 管理控制台。
2. 选择 Create database（创建数据库）。
3. 继续执行向导。记下为以下选项输入的值：
   - Master Username（主用户名）
   - Master Password（主密码）
4. 当您到达 Configure advanced settings（配置高级设置）时，对于 Network and Security（网络与安全性）设置，选择以下选项：
   - VPC – **`Not in VPC`**。如果此选项不可用，则您的账户可能不支持 EC2-Classic，或者您可能已选择仅在 VPC 中可用的实例类型。
   - Availability Zone（可用区）– **`No Preference`**
   - DB Security Group(s)（数据库安全组）–**`Create new Security Group`**
5. 配置剩余选项并选择 Create database（创建数据库）。记下为以下选项输入的值：
   - Database Name（数据库名称）
   - Database Port（数据库端口）

在 EC2-Classic 中，数据库实例具有数据库安全组，而不是 VPC 安全组。您无法将数据库安全组附加到 Elastic Beanstalk 环境。相反，您需要创建一个新安全组，您可为其授予访问数据库实例的权限并将其附加到环境。我们将此安全组称为桥接安全组 并将其命名为 **`webapp-bridge`**。

创建桥接安全组

1. 打开 Amazon EC2 控制台。
2. 在导航侧边栏中的 Network & Security（网络与安全性）下，选择 Security Groups（安全组）。
3. 选择 Create Security Group（创建安全组）。
4. 对于 Security group name（安全组名称），请键入 **`webapp-bridge`**。
5. 对于 Description（描述），键入 **`Provide access to DB instance from Elastic Beanstalk environment instances.`**。

6. 对于 VPC，将默认值保持选中状态。
7. 选择 Create（创建）。

接下来，修改附加到数据库实例的安全组以允许来自桥接安全组的入站流量。

修改 RDS 实例的安全组上的入口规则

1. 打开 Amazon RDS 控制台。
2. 选择 Databases（数据库）。
3. 选择您的数据库实例的名称以查看其详细信息。
4. 在 Connectivity（连接）部分中，在 Security（安全性）下，与数据库实例关联的安全组将会显示。打开链接以在 Amazon EC2 控制台中查看安全组。
5. 在安全组详细信息中，将 Connection Type（连接类型）设置为 EC2 Security Group（EC2 安全组）。
6. 将 EC2 Security Group Name（EC2 安全组名称）设置为您创建的桥接安全组的名称。
7. 选择 Authorize（授权）。

接下来，将桥接安全组添加到您的运行环境。此过程要求使用附加的其他安全组重新配置环境中的所有实例。

向环境添加安全组

- 请执行下列操作之一：
  - 使用 Elastic Beanstalk 控制台添加安全组

    a. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
    b. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

    > **Note**
    > 如果您有多个环境，请使用搜索栏筛选环境列表。

    c. 在导航窗格中，选择 Configuration（配置）。
    d. 在 Instances（实例）配置类别中，选择 Edit（编辑）。
    e. 在 EC2 security groups（EC2 安全组）下面，选择要附加到实例的安全组以及 Elastic Beanstalk 创建的实例安全组。

f. 要保存更改，请选择页面底部的 Apply（应用）。

g. 阅读警告，然后选择 Confirm（确认）。

- 要使用配置文件添加安全组，请使用 `securitygroup-addexisting.config` 示例文件。

接下来，使用环境属性将连接信息传递到环境。在使用 Elastic Beanstalk 控制台向环境中添加数据库实例时，Elastic Beanstalk 使用 RDS_HOSTNAME 等环境属性将连接信息传送到您的应用程序。您可以使用相同的属性，这将允许您将相同的应用程序代码与集成的数据库实例和外部的数据库实例结合使用，您也可以选择自己的属性名称。或者，您可以选择自己的属性名称。

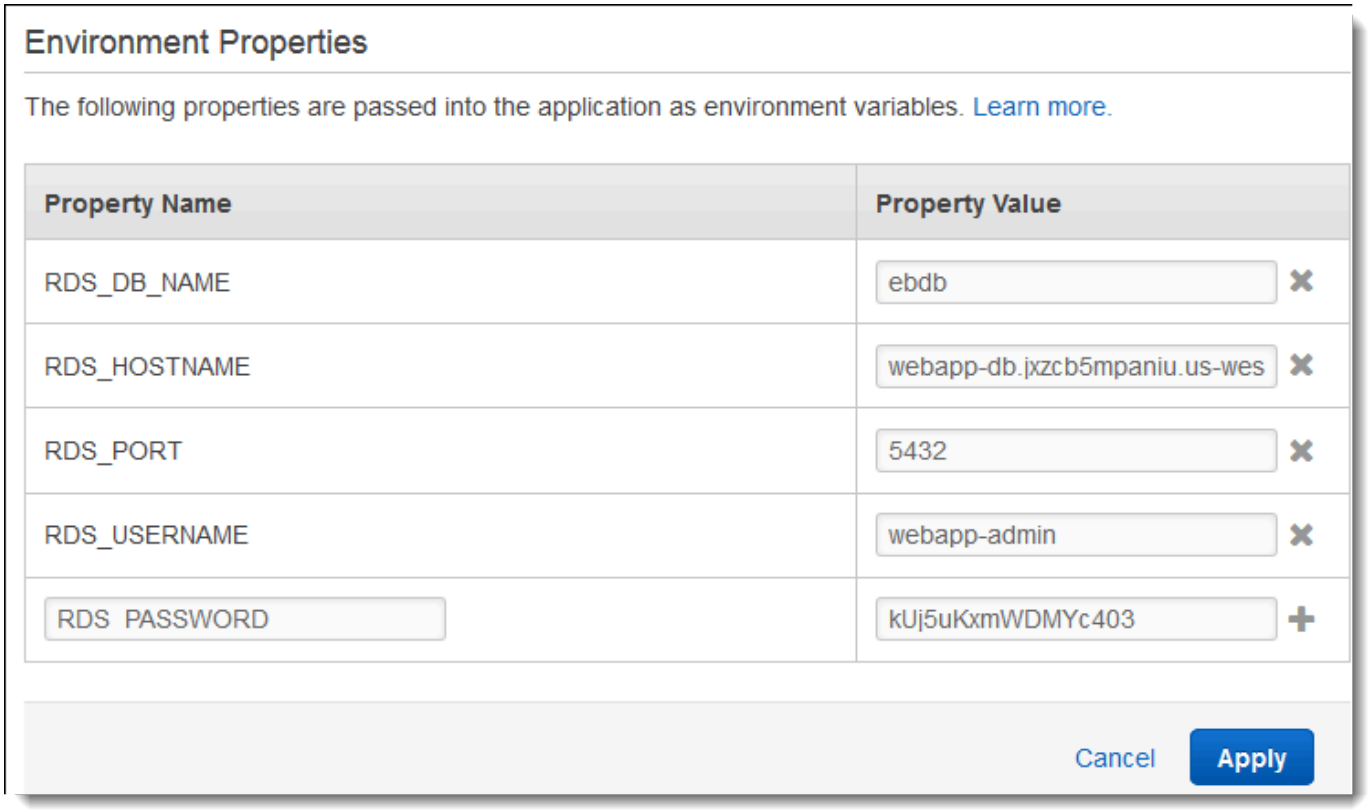
配置环境属性

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 在导航窗格中，选择 Configuration（配置）。
4. 在 Updates, monitoring, and logging（更新、监控和日志记录）配置类别中，选择 Edit（编辑）。
5. 在 Environment Properties（环境属性）部分中，定义应用程序读取的用于构建连接字符串的变量。为了与具有集成 RDS 数据库实例的环境兼容，请使用以下内容：

   - RDS_DB_NAME – Amazon RDS 控制台中的 DB Name（数据库名称）。
   - RDS_USERNAME – 向环境添加数据库时输入的 Master Username（主用户名）。
   - RDS_PASSWORD – 向环境添加数据库时输入的 Master Password（主密码）。
   - RDS_HOSTNAME – Amazon RDS 控制台中的数据库实例的 Endpoint（端点）。
   - RDS_PORT – Amazon RDS 控制台中的 Port（端口）。



6. 选择 Apply（应用）

如果您尚未将应用程序编程为读取环境属性和构建连接字符串，请参阅以下特定于语言的主题以获得指导：

- Java SE – 连接数据库 (Java SE 平台)
- Java with Tomcat – 连接数据库 (Tomcat 平台)
- Node.js – 连接到数据库
- .NET – 连接到数据库
- PHP – 使用 PDO 或 MySQLi 连接到数据库
- Python – 连接到数据库
- Ruby – 连接到数据库

最后，根据应用程序读取环境变量的时间，您可能需要在环境中的实例上重新启动应用程序服务器。

重新启动您的环境的应用程序服务器

1. 打开 Elastic Beanstalk 控制台，然后在 Regions（区域）列表中选择您的 AWS 区域。
2. 在导航窗格中，选择 Environments（环境），然后从列表中选择环境的名称。

> **Note**
>
> 如果您有多个环境，请使用搜索栏筛选环境列表。

3. 选择 Actions（操作），然后选择 Restart app server(s)（重启应用程序服务器）。

将 Amazon RDS 凭证存储在 AWS Secrets Manager 中

AWS Secrets Manager 通过提供存储和检索加密凭证的功能，帮助您改善安全状况。将凭证存储在 Secrets Manager 中有助于避免检查您应用程序或与其相关的组件的任何人泄露这些凭证。您的代码可以对 Secrets Manager 服务进行运行时调用，以动态检索凭证。Secrets Manager 还为运行时语言（包括 Python、Go 和 Java）提供客户端密钥缓存组件等功能。

有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的以下主题。

- Amazon RDS 如何使用 AWS Secrets Manager
- 创建 AWS Secrets Manager 数据库密钥
- 从 AWS Secrets Manager 中检索密钥

清除外部 Amazon RDS 实例

在将外部 Amazon RDS 实例连接到 Elastic Beanstalk 环境时，数据库实例不会与环境的生命周期相关联，因此不会在您终止环境时被删除。要确保可能已存储在数据库实例中的个人信息不会被不必要地保留，请删除不再需要的任何记录。或者，删除数据库实例。

将 Elastic Beanstalk 和 Amazon S3 结合使用

Amazon Simple Storage Service（Amazon S3）提供具备高持久性的容错型数据存储。

Elastic Beanstalk 为您在其中创建环境的每个区域创建一个名为 `elasticbeanstalk-region-account-id` 的 Amazon S3 存储桶。Elastic Beanstalk 使用此存储桶存储应用程序正常运行所需的对象，例如，临时配置文件。

Elastic Beanstalk 不会为其创建的 Amazon S3 存储桶启用默认加密。这意味着，在默认情况下，对象以未加密形式存储在存储桶中（并且只有授权用户可以访问）。有些应用程序需要在将所有对象存储在硬盘驱动器、数据库等中时对这些对象进行加密（也称为静态加密）。如果您有此要求，可以将您的账户的存储桶配置为默认加密。有关更多详细信息，请参阅《Amazon Simple Storage Service 用户指南》中的适用于 S3 存储桶的 Amazon S3 默认加密。

Elastic Beanstalk Amazon S3 存储桶的内容

下表列出了 Elastic Beanstalk 存储在您 `elasticbeanstalk-*` Amazon S3 存储桶中的一些对象。此表还显示了必须手动删除的对象。为避免产生不必要的存储成本，以及为确保不会保留个人信息，请务必在不再需要这些对象时手动删除它们。

Object	何时存储？	何时删除？
应用程序版本	创建环境或将应用程序代码部署到现有环境时，Elastic Beanstalk 会将应用程序版本存储在 Amazon S3 中并将此版本与环境关联。	删除应用程序期间，视版本生命周期而定。
源包	使用 Elastic Beanstalk 控制台或 EB CLI 上传新的应用程序版本时，Elastic Beanstalk 会将此版本的副本存储在 Amazon S3 中，并将其设置为环境的源包。	手动。删除应用程序版本时，可以选择从 Amazon S3 中删除版本以同时删除相关源包。有关详细信息，请参阅 管理应用程序版本。
自定义平台	创建自定义平台时，Elastic Beanstalk 会临时将相关数据存储在 Amazon S3 中。	自定义平台成功创建完成后。
日志文件	您可以请求 Elastic Beanstalk 检索实例日志文件（结尾或捆绑日志）并将它们存储在 Amazon S3 中。您还可启用日志轮换并将环境配置为在日志轮换后自动将日志发布到 Amazon S3。	结尾日志和捆绑日志：在创建后 15 分钟。 轮换日志：手动。
保存的配置	手动。	手动。

删除 Elastic Beanstalk Amazon S3 存储桶中的对象

在您终止环境或删除应用程序时，Elastic Beanstalk 将从 Amazon S3 中删除最相关的对象。为最大限度降低运行应用程序的存储成本，请定期删除应用程序不需要的对象。此外，请注意必须手动删除的对象，如 Elastic Beanstalk Amazon S3 存储桶的内容中所列出。要确保不会不必要地保留私有信息，请在不再需要这些对象时，将它们删除。

- 删除不希望再在应用程序中使用的应用程序版本。删除应用程序版本时，可以选择 Delete versions from Amazon S3（从 Amazon S3 中删除版本）以删除相关的源包，即应用程序源代码的副本和 Elastic Beanstalk 于部署应用程序或上传应用程序时上传到 Amazon S3 的配置文件。要了解如何删除应用程序版本，请参阅管理应用程序版本。
- 删除不需要的已轮换日志。或者，下载它们或将它们移动到 Amazon S3 Glacier 进行进一步分析。
- 删除不会再在任何环境中使用的已保存配置。

删除 Elastic Beanstalk Amazon S3 存储桶

当 Elastic Beanstalk 创建存储桶时，还会创建适用于新存储桶的存储桶策略。该策略有两个用途：

- 允许环境写入存储桶。
- 防止意外删除存储桶。

由于 Elastic Beanstalk 创建的策略适用于其为环境创建的存储桶，因此除非您有意先删除存储桶策略，否则您将无法删除这些存储桶。您可以从 Amazon S3 控制台中存储桶属性的权限部分删除存储桶策略。

> **警告**
>
> 如果删除 Elastic Beanstalk 已在您的账户中创建的某个存储桶，并且在相应区域仍具有现有应用程序和正在运行的环境，则您的应用程序可能停止正常运行。例如：
>
> - 当环境横向扩展时，Elastic Beanstalk 应能够在 Amazon S3 存储桶中找到环境的应用程序版本并使用它启动新的 Amazon EC2 实例。
> - 创建自定义平台时，Elastic Beanstalk 将在创建过程中使用临时 Amazon S3 存储。
>
> 建议从 Elastic Beanstalk Amazon S3 存储桶中删除不必要的特定对象，而不是删除整个存储桶。

删除 Elastic Beanstalk 存储桶（控制台）

《Amazon S3 用户指南》中的删除 S3 存储桶中还描述了删除 S3 存储桶的一般过程。由于接下来要在以下过程中删除由 Elastic Beanstalk 创建的存储桶，因此我们添加了其他步骤，首先删除存储桶策略。

1. 打开 Amazon S3 控制台。
2. 通过选择存储桶名称打开 Elastic Beanstalk 存储桶的页面。
3. 选择 Permissions（权限）选项卡。
4. 选择 Bucket Policy（存储桶策略）。
5. 选择 Delete (删除)。
6. 返回 Amazon S3 控制台主页，然后选择 Elastic Beanstalk 存储桶。
7. 选择 Delete Bucket（删除存储桶）。
8. 通过在文本字段中输入存储桶名称来确认要删除该存储桶，然后选择删除存储桶。

将 Elastic Beanstalk 和 Amazon VPC 结合使用

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 为 Elastic Beanstalk 应用程序和相关AWS资源创建安全网络。在创建您的环境时，您可以选择用于您的应用程序实例和负载均衡器的 VPC、子网和安全组。您可以使用所需的任何 VPC 配置，只要它满足以下要求即可。

VPC 要求

- Internet 访问 - 实例可以通过下列方法之一访问 Internet：
  - 公有子网 - 实例具有公有 IP 地址并使用互联网网关访问 Internet。
  - 私有子网 - 实例使用 NAT 设备访问 Internet。

> **Note**
>
> 如果您将 VPC 中的 VPC 终端节点配置为同时连接到 `elasticbeanstalk` 和 `elasticbeanstalk-health` 服务，则 Internet 访问是可选的，并且仅在应用程序特别需要时才需要它。如果没有 VPC 终端节点，则您的 VPC 必须具有 Internet 访问权限。Elastic Beanstalk 为您设置的默认 VPC 提供了 Internet 访问权限。

Elastic Beanstalk 不支持使用代理设置（如 `HTTPS_PROXY`）配置 Web 代理。

- NTP - Elastic Beanstalk 环境中的实例使用网络时间协议 (NTP) 同步系统时钟。如果实例在 UDP 端口 123 上无法通信，时钟将无法同步，从而导致问题并发出 Elastic Beanstalk 运行状况报告。请确保您的 VPC 安全组和网络 ACL 在端口 123 上允许入站和出站 UDP 流量以避免这些问题。

elastic-beanstalk-samples 存储库提供了 AWS CloudFormation 模板，可使用这些模板创建 VPC 以用于 Elastic Beanstalk 环境。

使用 AWS CloudFormation 模板创建资源

1. 克隆示例存储库或使用自述文件中的链接下载模板。
2. 打开 AWS CloudFormation 控制台。
3. 选择创建堆栈。
4. 选择将模板上传到 Amazon S3。
5. 选择上传文件，然后从本地计算机中上传模板文件。
6. 选择下一步，然后按照说明使用模板中的资源创建堆栈。

在堆栈创建完成后，检查 Outputs (输出) 选项卡以查找 VPC ID 和子网 ID。可以使用这些 ID 在新建环境向导的网络配置类别中配置 VPC。

主题

公有 VPC

AWS CloudFormation 模板 – vpc-public.yaml

设置 (负载均衡)

- 负载均衡器可见性 - 公有

- 负载均衡器子网 - 两个公有子网
- 实例公有 IP - 已启用
- 实例子网 - 两个公有子网
- 实例安全组 - 添加默认安全组

设置 (单个实例)

- 实例子网 - 公有子网之一
- 实例安全组 - 添加默认安全组

基本仅公有 VPC 布局包含一个或多个公有子网，一个互联网网关和一个默认安全组（允许 VPC 中的资源之间的流量）。在 VPC 中创建环境时，Elastic Beanstalk 会创建额外的资源（因环境类型而异）。

VPC 资源

- 单一实例 - Elastic Beanstalk 为应用程序实例创建一个安全组以允许端口 80 上的 Internet 流量，并为实例分配一个弹性 IP 以为其提供公有 IP 地址。环境的域名将解析为实例的公有 IP 地址。
- 负载均衡 - Elastic Beanstalk 为负载均衡器创建一个安全组以允许端口 80 上的 Internet 流量，并为应用程序实例创建一个安全组以允许来自负载均衡器的安全组的流量。环境的域名将解析为负载均衡器的公有域名。

这与在使用默认 VPC 时 Elastic Beanstalk 管理网络的方式类似。公有子网中的安全性取决于 Elastic Beanstalk 创建的负载均衡器和实例安全组。它是成本最低的配置，因为它不需要使用 NAT 网关。

公有/私有 VPC

AWS CloudFormation 模板 – vpc-privatepublic.yaml

设置 (负载均衡)

- 负载均衡器可见性 - 公有
- 负载均衡器子网 - 两个公有子网
- 实例公有 IP - 已禁用
- 实例子网 - 两个私有子网

- 实例安全组 - 添加默认安全组

要提高安全性，请在您的 VPC 中添加私有子网以创建公有-私有 布局。该布局需要在公有子网中使用负载均衡器和 NAT 网关，并允许您在私有子网中运行您的应用程序实例、数据库和任何其他资源。私有子网中的实例只能通过负载均衡器和 NAT 网关与 Internet 通信。

私有 VPC

AWS CloudFormation 模板 – vpc-private.yaml

设置 (负载均衡)

- 负载均衡器可见性 - 私有
- 负载均衡器子网 - 两个私有子网
- 实例公有 IP - 已禁用
- 实例子网 - 两个私有子网
- 实例安全组 - 添加默认安全组

对于不应从 Internet 访问的内部应用程序，您可以在私有子网中运行所有内容，并将负载均衡器配置为面向内部（将 Load balancer visibility (负载均衡器可见性) 更改为 Internal (内部)）。此模板创建一个没有公有子网和互联网网关的 VPC。可以将该布局用于只应从同一 VPC 或附加的 VPN 中访问的应用程序。

在私有 VPC 中运行 Elastic Beanstalk 环境

如果在私有 VPC 中创建 Elastic Beanstalk 环境，则该环境无法访问 Internet。您的应用程序可能需要访问 Elastic Beanstalk 服务或其他服务。您的环境可能使用增强型运行状况报告，在此情况下，环境实例会将运行状况信息发送到增强型运行状况服务。环境实例上的 Elastic Beanstalk 代码将流量发送到其他AWS服务，并将其他流量发送到非AWS端点（例如，为您的应用程序下载依赖项包）。以下是您在此情况下为确保环境正常运行而可能需要执行的一些步骤。

- 为 Elastic Beanstalk 配置 VPC 终端节点 - Elastic Beanstalk 以及增强型运行状况服务支持 VPC 终端节点，这将确保流向这些服务的流量保持在 Amazon 网络内并且不需要 Internet 访问权限。有关更多信息，请参阅 the section called “VPC 终端节点”。
- 为其他服务配置 VPC 终端节点 – Elastic Beanstalk 代表您将流量发送到多项其他的AWS服务：Amazon Simple Storage Service (Amazon S3)、Amazon Simple Queue Service (Amazon

SQS)、AWS CloudFormation 和 Amazon CloudWatch Logs。您也必须为这些服务配置 VPC 终端节点。有关 VPC 终端节点（包括每个服务链接）的详细信息，请参阅 Amazon VPC 用户指南 中的 VPC 终端节点。

> **Note**
>
> 某些AWS服务（包括 Elastic Beanstalk）在有限数量的AWS区域中支持 VPC 终端节点。在设计私有 VPC 解决方案时，请确认 Elastic Beanstalk 以及此处提及的其他依赖服务支持您选择的AWS区域中的 VPC 终端节点。

- 提供私有 Docker 映像 - 在 Docker 环境中，环境实例上的代码可能会尝试在环境创建期间从 Internet 中提取配置的 Docker 映像，但此操作将失败。要避免发生此失败，请在您的环境中构建自定义 Docker 映像，或使用存储在 Amazon Elastic Container Registry (Amazon ECR) 中的 Docker 映像并为 Amazon ECR 服务配置 VPC 终端节点。
- 启用 DNS 名称 – 环境实例上的 Elastic Beanstalk 代码使用其公有端点将流量发送到所有AWS服务。要确保此流量通过，请在配置所有接口 VPC 终端节点时选择 Enable DNS name (启用 DNS 名称) 选项。这会在您的 VPC 中添加一个 DNS 条目，该条目将公有服务终端节点映射到接口 VPC 终端节点。

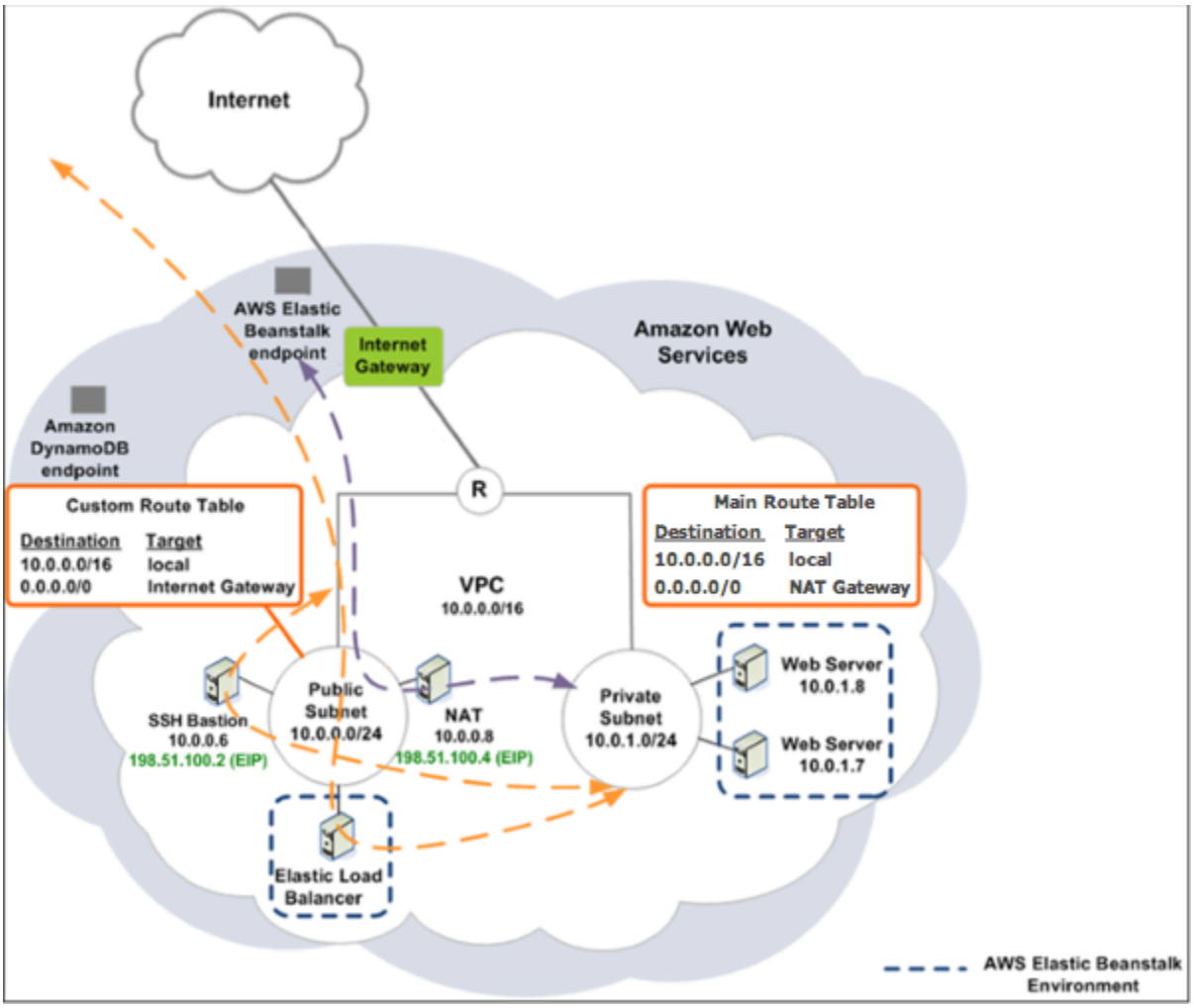
> **Important**
>
> 如果您的 VPC 不是私有的，并且具有公有 Internet 访问权限，并且如果为任何 VPC 终端节点禁用了 Enable DNS name (启用 DNS 名称)，则流向相应服务的流量将通过公有 Internet 传输。这可能不是您期望实现的。使用私有 VPC 可轻松检测到此问题，因为它可以防止此流量通过并且您会收到错误。但是，对于公有 VPC，您不会收到任何指示。

- 包含应用程序依赖项 - 如果您的应用程序具有诸如语言运行时包之类的依赖项，它可能会尝试在环境创建期间从 Internet 下载并安装这些依赖项，但此操作将失败。要避免发生此失败，请在应用程序的源包中包含所有依赖项包。
- 使用当前平台版本 - 请确保您的环境使用 2020 年 2 月 24 日版或更高版本的平台。具体而言，请使用在以下两个更新之一中或之后发布的平台版本：Linux Update 2020-02-28、Windows Update 2020-02-24。

> **Note**
>
> 需要更新的平台版本的原因是，旧版本存在一个问题，此问题可能会阻止通过 Enable DNS name (启用 DNS 名称) 选项创建的 DNS 条目正常用于 Amazon SQS。

示例：使用堡垒主机启动 VPC 中的 Elastic Beanstalk 应用程序

如果您的 Amazon EC2 实例位于私有子网内，则将无法远程连接到这些实例。要连接到实例，您可以在公有子网中设置堡垒服务器以作为代理。例如，您可以在公有子网中设置 SSH 端口转发器或 RDP 网关，以代理从您自己的网络流向数据库服务器的数据流。本节提供有关如何创建包含私有和公有子网的 VPC 的示例。实例位于私有子网内，而堡垒主机、NAT 网关和负载均衡器位于公有子网内。您的基础设施与下图类似。



要使用堡垒主机在 VPC 内部署 Elastic Beanstalk 应用程序，请完成以下小节中描述的步骤。

步骤

- 创建带有公有子网和私有子网的 VPC
- 创建和配置堡垒主机安全组
- 更新实例安全组
- 创建堡垒主机

创建带有公有子网和私有子网的 VPC

完成公有/私有 VPC中的所有过程。在部署应用程序时，您必须为实例指定 Amazon EC2 键对，以便可远程连接到这些实例。有关如何指定实例键对的更多信息，请参阅您的 Elastic Beanstalk 环境的 Amazon EC2 实例。

创建和配置堡垒主机安全组

创建堡垒主机的安全组并添加规则，该规则允许来自 Internet 的入站 SSH 流量以及流向包含 Amazon EC2 实例的私有子网的出站 SSH 流量。

创建堡垒主机安全组

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Security Groups。
3. 选择 Create Security Group。
4. 在 Create Security Group (创建安全组) 对话框中，输入以下内容并选择 Yes, Create (是，创建)。

   Name tag (名称标签)（可选）

   输入安全组的名称标签。

   组名：

   输入安全组的名称。

   描述

   输入安全组的描述。

   VPC：

   选择您的 VPC。

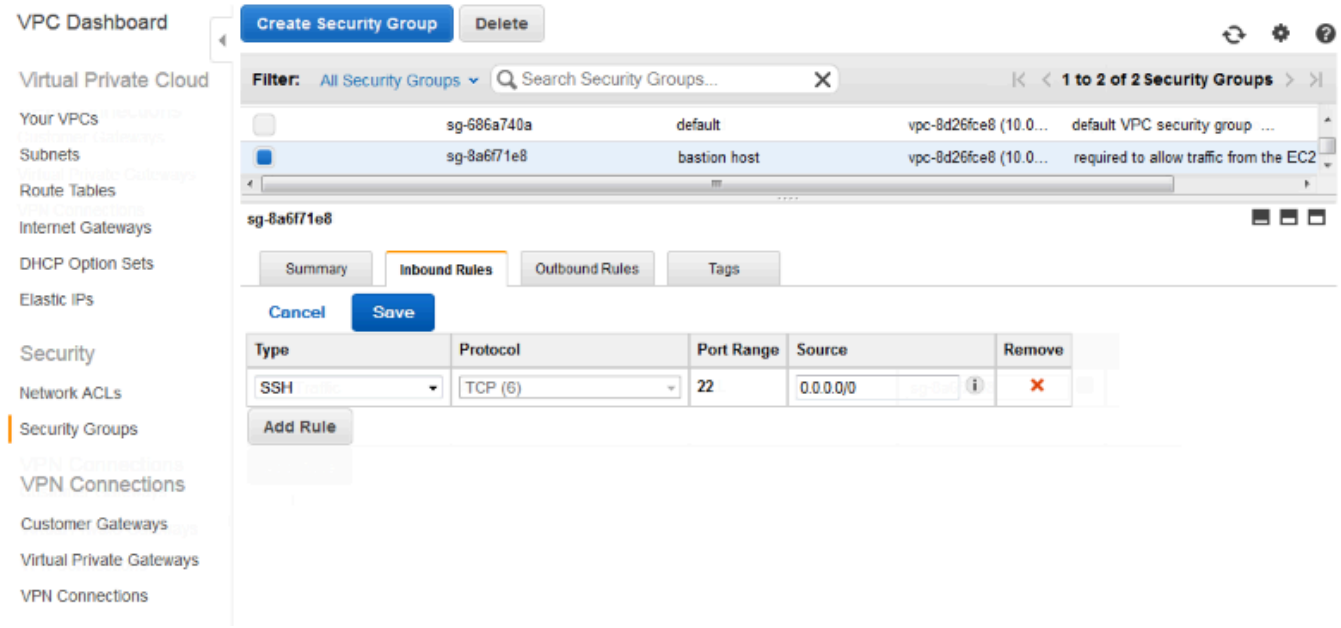
   此时将创建安全组并在 Security Groups (安全组) 页上显示它。请注意，它具有 ID（例如，sg-xxxxxxxx）。您可能必须通过单击该页面右上角的 Show/Hide (显示/隐藏) 来启用 Group ID (组 ID) 列。

配置堡垒主机安全组

1. 在安全组列表中，选中刚才为堡垒主机创建的安全组的复选框。

2. 在 Inbound Rules 选项卡上，选择 Edit。
3. 如果需要，请选择 Add another rule (再添加一条规则)。
4. 如果堡垒主机是 Linux 实例，请选择 Type (类型) 下的 SSH。

   如果堡垒主机是 Windows 实例，请选择 Type (类型) 下的 RDP。
5. 在 Source (源) 字段中输入所需的源 CIDR 范围，然后选择 Save (保存)。

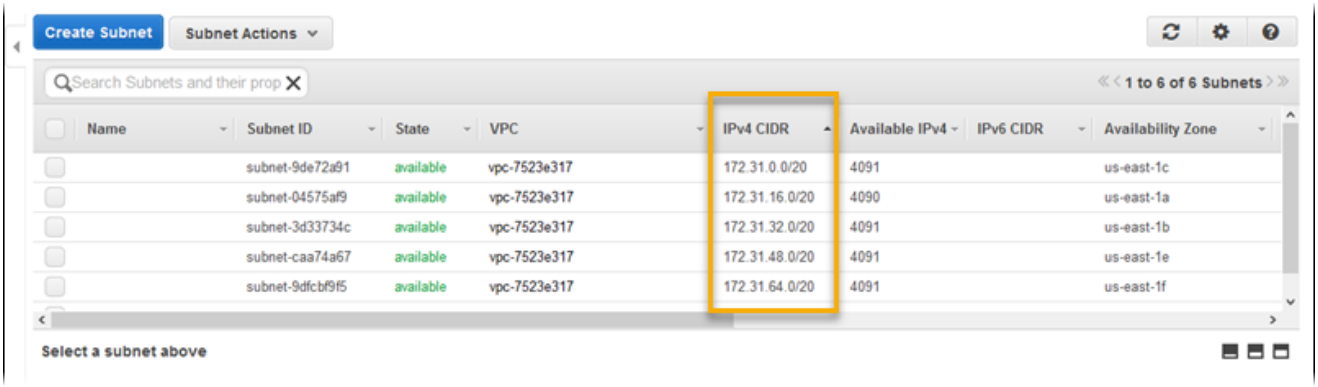


6. 在 Outbound Rules (出站规则) 选项卡上，选择 Edit (编辑)。
7. 如果需要，请选择 Add another rule (再添加一条规则)。
8. 在 Type (类型) 下，选择为入站规则指定的类型。
9. 在 Source (源) 字段中，输入主机的子网在 VPC 的私有子网中的 CIDR 范围。

   要查找它：

   a. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
   b. 在导航窗格中，选择 Subnets (子网)。
   c. 对于您希望防御主机桥接到的主机，记下包含后者的每个 Availability Zone (可用区) 的 IPv4 CIDR 下的值。

> **Note**
>
> 如果您在多个可用区中有主机，请为每个可用区创建一个出站规则。



10. 选择保存。

更新实例安全组

默认情况下，您为实例创建的安全组不允许传入流量。尽管 Elastic Beanstalk 会修改实例的默认组来允许 SSH 流量时，但如果您的实例是 Windows 实例，您必须修改自定义实例安全组来允许 RDP 流量。

更新 RDP 的实例安全组

1. 在安全组的列表中，选中实例安全组的复选框。
2. 在 Inbound (入站) 选项卡上，选择 Edit (编辑)。
3. 如果需要，请选择 Add another rule (再添加一条规则)。
4. 输入下列值并选择 Save (保存)。

   类型

   RDP

   Protocol（协议）– 。

   TCP

   端口范围

   3389

   源

   输入堡垒主机安全组的 ID（例如，`sg-8a6f71e8`）并选择 Save (保存)。

创建堡垒主机

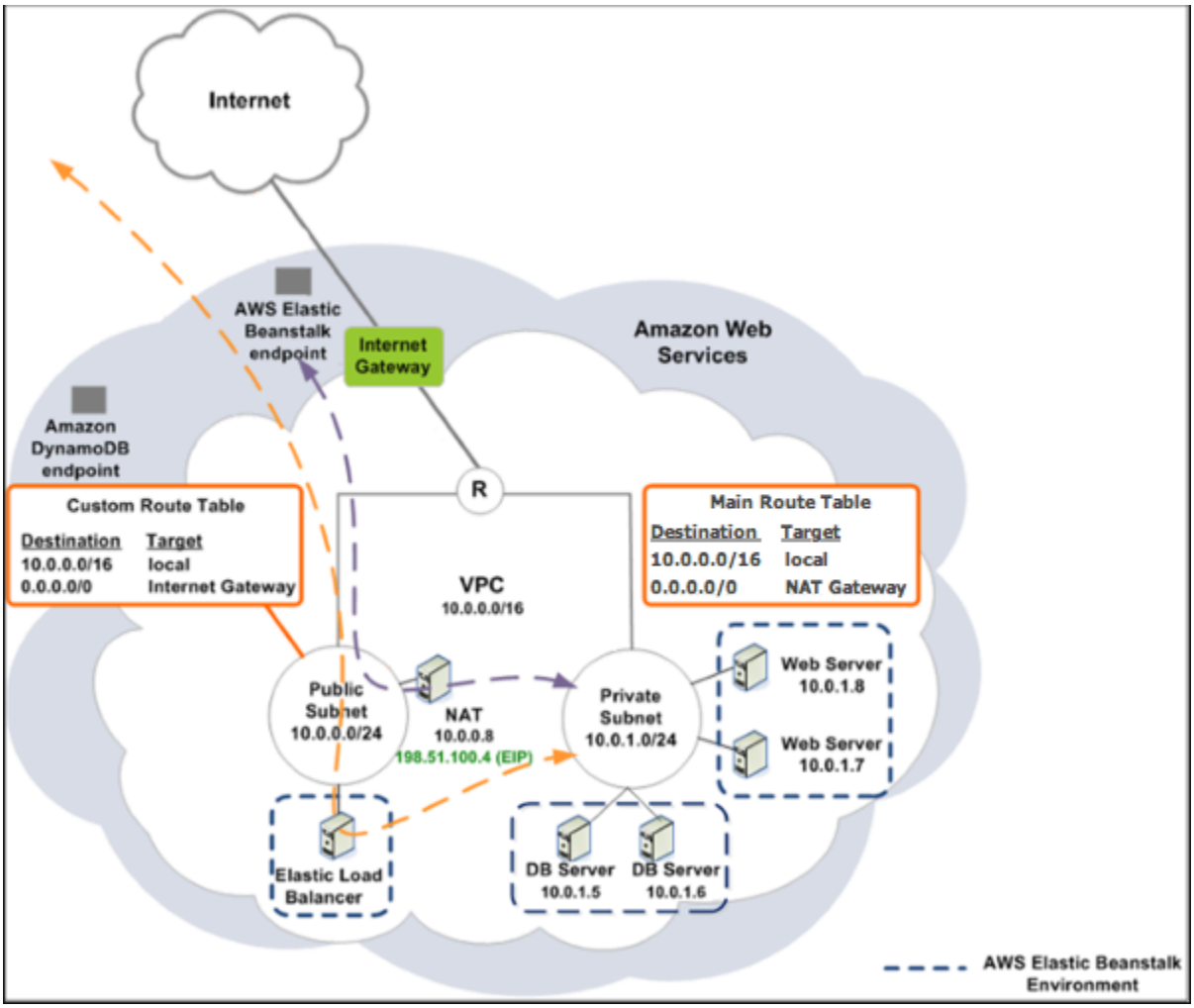
为了创建堡垒主机，您应在充当堡垒主机的公有子网中启动 Amazon EC2 实例。

有关在私有子网中设置适用于 Windows 实例的堡垒主机的更多信息，请参阅使用堡垒服务器控制对 EC2 实例的网络访问。

有关在私有子网中设置适用于 Linux 实例的堡垒主机的更多信息，请参阅安全地连接到在私有 Amazon VPC 中运行的 Linux 实例。

示例：使用 Amazon RDS 启动 VPC 中的 Elastic Beanstalk

此部分演示了使用 NAT 网关在 VPC 中部署带 Amazon RDS 的 Elastic Beanstalk 应用程序的任务。您的基础设施与下图类似。



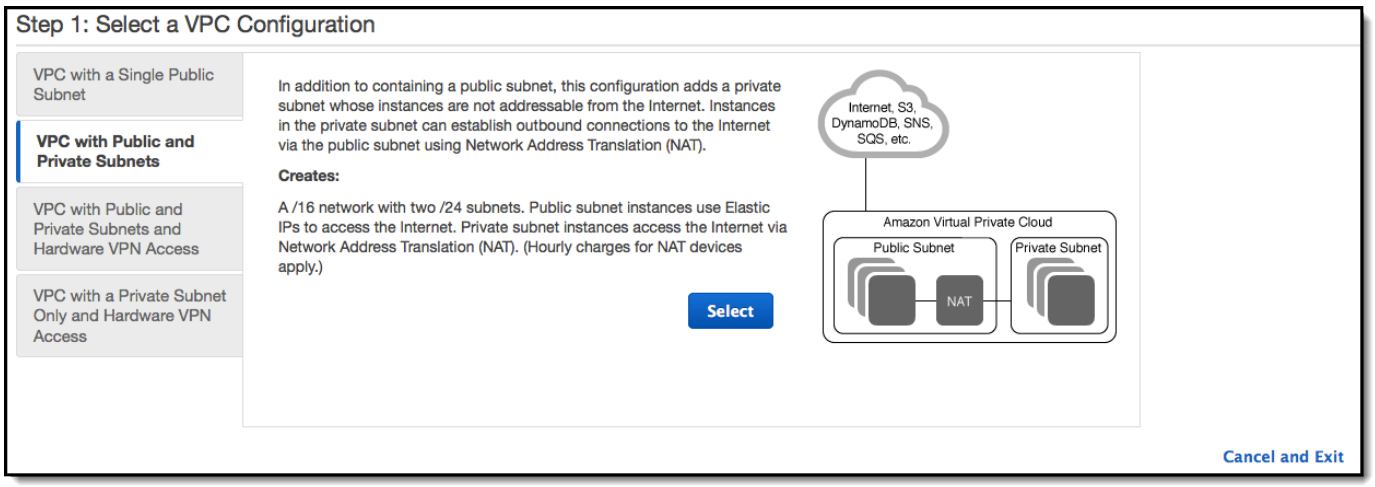
> **Note**
>
> 如果您之前从未和应用程序同时使用数据库实例，请在添加 VPC 配置至组合之前，尝试添加至测试环境和连接到外部数据库实例。

创建带有公有子网和私有子网的 VPC

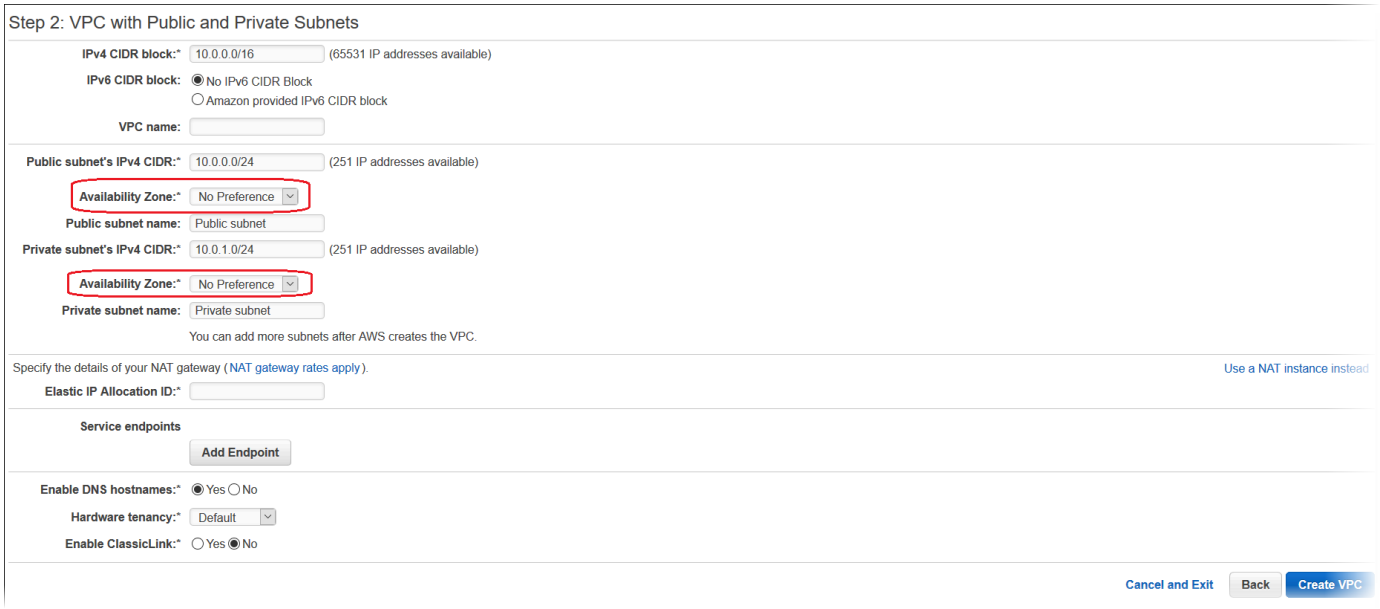
您可使用 Amazon VPC 控制台创建 VPC。

创建 VPC

1. 登录 Amazon VPC 控制台。
2. 在导航窗格中，选择 VPC Dashboard (VPC 控制面板)。然后选择 Create VPC (创建 VPC)。
3. 选择 VPC with Public and Private Subnets (带有公有子网和私有子网的 VPC)，然后选择 Select (选择)。



4. 您的 Elastic Load Balancing 负载均衡器和您的 Amazon EC2 实例必须位于同一可用区中，这样它们才能相互通信。从每个 Availability Zone (可用区) 列表中选择相同的可用区。

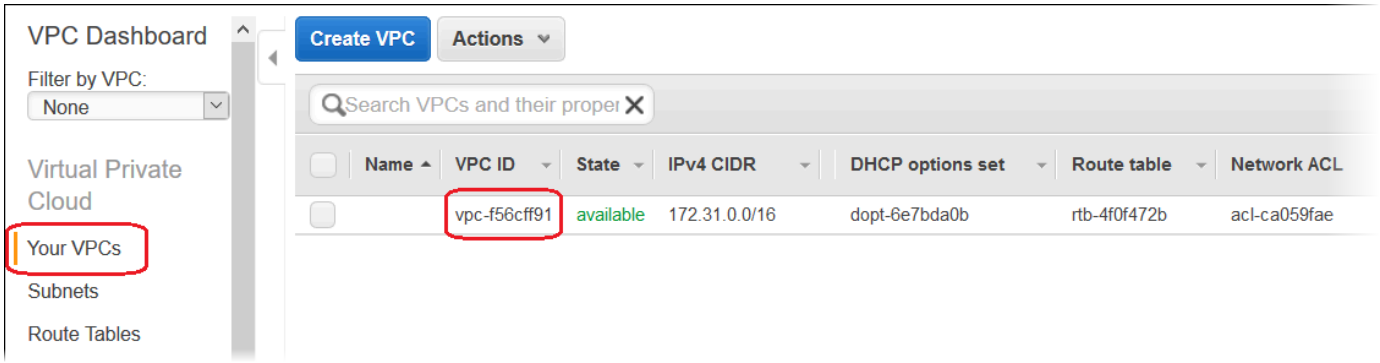


5. 为您的 NAT 网关选择弹性 IP 地址。
6. 选择 Create VPC (创建 VPC)。

   向导将开始创建 VPC、子网和 Internet 网关。它还将更新主路由表并创建自定义路由表。最后，向导将在公有子网中创建 NAT 网关。

> **Note**
>
> 您可以选择在公有子网而非 NAT 网关中启动 NAT 实例。有关更多信息，请参阅 Amazon VPC 用户指南 中的场景 2：带有公有子网和私有子网的 VPC (NAT)。

7. 在成功创建 VPC 后，您会获得一个 VPC ID。在下一个步骤中，您需要用到此值。要查看您的 VPC ID，请在 Amazon VPC 控制台的左窗格中选择 Your VPCs (您的 VPC)。

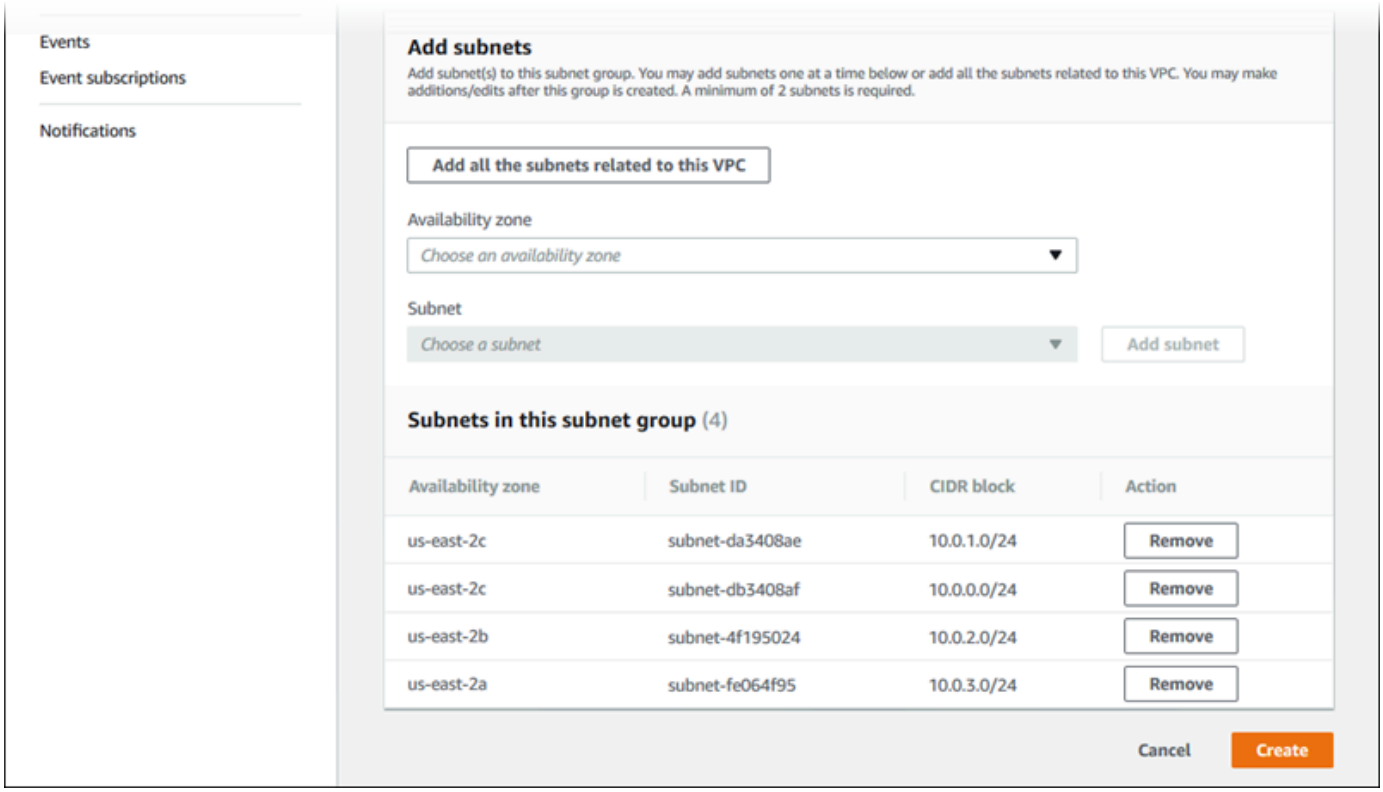


创建数据库子网组

VPC 的数据库子网组是可为后端 RDS 数据库实例指定的子网（通常为私有子网）集合。每个数据库子网组应至少包含给定 AWS 区域中每个可用区的一个子网。要了解更多信息，请参阅在 VPC 中创建子网。

创建数据库子网组

1. 打开 Amazon RDS 控制台。
2. 在导航窗格中，选择子网组。
3. 选择创建数据库子网组。
4. 选择名称，然后键入数据库子网组的名称。
5. 选择描述，然后描述数据库子网组。
6. 对于 VPC，选择您之前创建的 VPC 的 ID。
7. 在添加子网中，选择添加与此 VPC 相关的所有子网。



8. 完成后，选择 Create。

   您的新数据库子网组显示在 Amazon RDS 控制台的子网组列表中。在此页面底部的详细信息窗格中，您可以选择它以查看详细信息，例如与此组关联的所有子网。

部署到 Elastic Beanstalk

设置 VPC 后，您可在其中创建您的环境并将应用程序部署到 Elastic Beanstalk。您可以使用 Elastic Beanstalk 控制台执行此操作，也可以使用AWS工具包、AWS CLI、EB CLI 或 Elastic Beanstalk API。如果您使用 Elastic Beanstalk 控制台，则只需上传 `.war` 或 `.zip` 文件并在向导中选择 VPC 设置。Elastic Beanstalk 随后会在 VPC 中创建环境并部署应用程序。或者，您可以使用AWS工具包、AWS CLI、EB CLI 或 Elastic Beanstalk API 来部署应用程序。为此，您需要在配置文件中定义 VPC 选项设置并使用源捆绑部署此文件。本主题提供了这两种方法的说明。

使用 Elastic Beanstalk 控制台来部署

Elastic Beanstalk 控制台将指导您在您的 VPC 内创建新环境。您需要提供 `.war` 文件 (对于 Java 应用程序) 或 `.zip` 文件 (对于所有其他应用程序)。在 Elastic Beanstalk 环境向导的 VPC Configuration (VPC 配置) 页面上，您必须做出以下选择：

VPC：

选择您的 VPC。

VPC security group (VPC 安全组)。

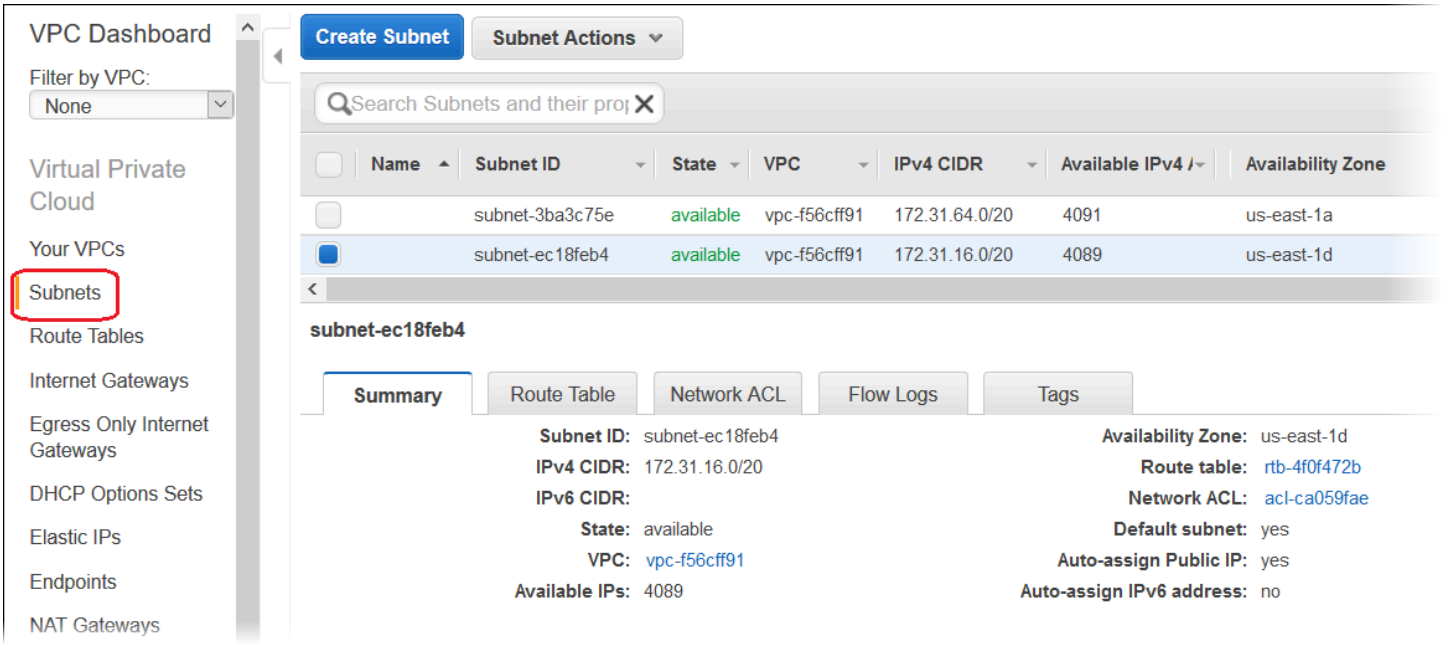
选择您上面创建的实例安全组。

ELB visibility (ELB 可见性

选择 `External` (如果您的负载均衡器应公开可用)，或选择 `Internal` (如果负载均衡器应仅在您的 VPC 内可用)。

选择您的负载均衡器和 EC2 实例的子网。确保选择负载均衡器的公有子网和 Amazon EC2 实例的私有子网。默认情况下，VPC 创建向导将创建 `10.0.0.0/24` 形式的公有子网，并创建 `10.0.1.0/24` 形式的私有子网。

您可以通过在 Amazon VPC 控制台中选择 Subnets (子网) 来查看子网 ID。



使用AWS工具包、EB CLI、AWS CLI 或 API 进行部署

使用AWS工具包、EB CLI、AWS CLI 或 API 将应用程序部署到 Elastic Beanstalk 时，您可以在一个文件中指定 VPC 选项设置并使用源代码包部署该文件。参阅 使用配置文件 (.ebextensions) 进行高级环境自定义 了解更多信息。

更新选项设置时，至少需要指定以下内容：

- VPCId - 包含 VPC 的 ID。
- Subnets (子网) – 包含 Auto Scaling 组子网的 ID。在此示例中，这是私有子网的 ID。
- ELBSubnets - 包含负载均衡器的子网 ID。在此示例中，这是公有子网的 ID。
- SecurityGroups - 包含安全组的 ID。
- DBSubnets - 包含数据库子网的 ID。

> **Note**
>
> 使用数据库子网时，您需要在 VPC 中创建其他子网，以覆盖该 AWS 区域中的所有可用区。

另外，您还可以指定以下信息：

- ELBScheme - 指定 `internal` 以在 VPC 内创建一个内部负载均衡器，确保无法从 VPC 外部访问您的 Elastic Beanstalk 应用程序。

以下是在 VPC 内部署 Elastic Beanstalk 应用程序时可以使用的选项设置的示例。有关 VPC 选项设置的更多信息（包括有关如何指定这些设置、默认值和有效值的示例），请参阅配置选项中的 aws:ec2:vpc 命名空间表。

```
option_settings:
  - namespace: aws:autoscaling:launchconfiguration
    option_name: EC2KeyName
    value: ec2keypair

  - namespace: aws:ec2:vpc
    option_name: VPCId
    value: vpc-170647c

  - namespace: aws:ec2:vpc
    option_name: Subnets
    value: subnet-4f195024

  - namespace: aws:ec2:vpc
    option_name: ELBSubnets
    value: subnet-fe064f95

  - namespace: aws:ec2:vpc
    option_name: DBSubnets
    value: subnet-fg148g78

  - namespace: aws:autoscaling:launchconfiguration
    option_name: InstanceType
    value: m1.small

  - namespace: aws:autoscaling:launchconfiguration
    option_name: SecurityGroups
    value: sg-7f1ef110
```

> **Note**
>
> 使用数据库子网时，确保 VPC 中的子网可以覆盖该 AWS 区域中的所有可用区。

将 Elastic Beanstalk 与 VPC 终端节点结合使用

VPC 终端节点 使您能够将 VPC 私密地连接到支持的 AWS 服务和 VPC 终端节点服务（由 AWS PrivateLink 提供支持），而无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。

VPC 中的实例无需公有 IP 地址便可与服务中的资源进行通信。VPC 和其他服务之间的流量不会脱离 Amazon 网络。有关 VPC 终端节点的完整信息，请参阅 Amazon VPC 用户指南 中的 VPC 终端节点。

AWS Elastic Beanstalk 支持 AWS PrivateLink，这将提供与 Elastic Beanstalk 服务的私有连接并使流量远离公有互联网。要使您的应用程序能够使用 AWS PrivateLink 向 Elastic Beanstalk 发送请求，您可以配置一类称为接口 VPC 终端节点（接口端点）的 VPC 终端节点。有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

> **Note**
>
> Elastic Beanstalk 在有限数量的AWS区域中支持 AWS PrivateLink 和接口 VPC 终端节点。我们正在努力在不久的将来向更多 AWS 区域提供支持。

为 Elastic Beanstalk 设置 VPC 终端节点

要在 VPC 中为 Elastic Beanstalk 服务创建接口 VPC 终端节点，请遵循创建接口终端节点过程。对于 Service Name (服务名称)，请选择 com.amazonaws.*region*.elasticbeanstalk。

如果已为 VPC 配置公有 Internet 访问权限，则您的应用程序仍可使用 `elasticbeanstalk.`*region*`.amazonaws.com` 公有终端节点通过 Internet 访问 Elastic Beanstalk。可以通过确保在终端节点创建期间启用 Enable DNS name (启用 DNS 名称) 来防止这种情况（默认情况下为 true）。这会在您的 VPC 中添加一个 DNS 条目，该条目将公有服务终端节点映射到接口 VPC 终端节点。

为增强型运行状况设置 VPC 终端节点

如果您为环境启用了增强型运行状况报告，则也可将增强型运行状况信息配置为通过 AWS PrivateLink 发送。增强型运行状况信息由 `healthd` 守护程序（环境实例上的 Elastic Beanstalk 组件）发送到单独的 Elastic Beanstalk 增强型运行状况服务。要在 VPC 中为此服务创建接口 VPC 终端节点，请遵循创建接口终端节点过程。对于 Service Name (服务名称)，请选择 com.amazonaws.*region*.elasticbeanstalk-health。

> **Important**
>
> `healthd` 守护程序将增强型运行状况信息发送到公有终端节点 `elasticbeanstalk-health.`*region*`.amazonaws.com`。如果已为 VPC 配置公有 Internet 访问权限，并且为 VPC 终端节点禁用了 Enable DNS name (启用 DNS 名称)，则增强型运行状况信息将通过公

有 Internet 传播。在设置增强型运行状况 VPC 终端节点时，这可能不是您的意图。请确保启用 Enable DNS name (启用 DNS 名称) (默认情况下为 true) 。

在私有 VPC 中使用 VPC 终端节点

私有 VPC 或 VPC 中的私有子网没有公有 Internet 访问权限。您可能希望在私有 VPC 中运行 Elastic Beanstalk 环境并配置接口 VPC 终端节点以增强安全性。在此情况下，请注意，除了联系 Elastic Beanstalk 服务之外，您的环境可能会出于其他原因尝试连接到 Internet。要了解有关在私有 VPC 中运行环境的更多信息，请参阅the section called “在私有 VPC 中运行 Elastic Beanstalk 环境”。

使用终端节点策略控制通过 VPC 终端节点进行的访问

默认情况下，VPC 终端节点允许对与其关联的服务进行完全访问。创建或修改终端节点时，可以将终端节点策略 附加到终端节点。

端点策略是一种 AWS Identity and Access Management (IAM) 资源策略，用于通过端点控制对指定服务的访问。终端节点策略特定于某个终端节点。它独立于您的环境可能拥有的任何用户或实例 IAM 策略，并且不会覆盖或替换这些策略。有关创作和使用 VPC 终端节点策略的详细信息，请参阅 Amazon VPC 用户指南 中的使用 VPC 终端节点控制对服务的访问。

以下示例拒绝所有用户通过 VPC 终端节点终止环境的权限，并允许对所有其他操作进行完全访问。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "elasticbeanstalk:TerminateEnvironment",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

> **Note**
>
> 目前，仅主 Elastic Beanstalk 服务支持将终端节点策略附加到其 VPC 终端节点。增强型运行状况服务不支持终端节点策略。

配置用于 Elastic Beanstalk 的开发计算机

该页面说明了如何设置本地计算机以开发 AWS Elastic Beanstalk 应用程序。它介绍了文件夹结构、源代码管理和 CLI 工具。

主题

创建项目文件夹

为您的项目创建一个文件夹。您可以将此文件夹存储在您拥有读写权限的任意本地磁盘位置。在您的用户文件夹中创建文件夹也可以。如果您计划处理多个应用程序，请在一个名称类似 `workspace` 或 `projects` 的文件夹中创建项目文件夹，以方便组织和管理：

```
workspace/
|-- my-first-app
`-- my-second-app
```

项目文件夹的内容根据应用程序所使用的 Web 容器或框架而不同。

> **Note**
>
> 避免使用在文件夹名称或任何路径元素中包含单引号 (') 或双引号 (") 字符的文件夹和路径。如果文件夹名称中包含任意上述字符，一些 Elastic Beanstalk 命令会运行失败。

设置源代码控制

设置源代码控制可针对意外删除项目文件夹中的文件或代码提供保护，还可以恢复造成项目中断的更改。

如果您没有源代码控制系统，可以考虑使用 Git，它是一种很好用的免费工具，与 Elastic Beanstalk 命令行界面 (CLI) 集成良好。请访问 Git 主页安装 Git。

请按照 Git 网站上的说明安装和配置 Git，然后在您的项目文件夹中运行 `git init` 以设置本地存储库。

```
~/workspace/my-first-app$ git init
Initialized empty Git repository in /home/local/username/workspace/my-first-app/.git/
```

当您向项目文件夹添加内容或更新内容时，请将更改提交到 Git 存储库：

```
~/workspace/my-first-app$ git add default.jsp
~/workspace/my-first-app$ git commit -m "add default JSP"
```

每次提交时，都会创建一个项目快照，在以后发生错误时可以还原。有关 Git 命令和工作流程的更多信息，请参阅 Git 文档。

配置远程存储库

若您的硬盘驱动器崩溃，或您希望在其他计算机上处理您的项目，该怎么办？ 要在线备份您的源代码并从任意计算机进行访问，请配置一个可将提交的内容推送到其中的远程存储库。

借助 AWS CodeCommit，您可在AWS云中创建私有存储库。CodeCommit 包含在AWS免费套餐中，可供账户中的最多五个 AWS Identity and Access Management (IAM) 用户使用。 有关定价详细信息，请参阅 AWS CodeCommit 定价。

有关设置说明，请访问 AWS CodeCommit 用户指南。

GitHub 是在线存储项目代码的另一个常用选择。它允许您免费创建公有在线存储库，也支持按月收费的私有存储库。请到 github.com 注册获取 GitHub。

为项目创建远程存储库以后，请使用 `git remote add` 将其附加到您的本地存储库：

```
~/workspace/my-first-app$ git remote add origin ssh://git-codecommit.us-
east-2.amazonaws.com/v1/repos/my-repo
```

安装 EB CLI

使用 EB CLI 可从命令行管理您的 Elastic Beanstalk 环境并监控运行状况。有关安装说明，请参阅安装 EB CLI。

默认情况下，EB CLI 将您的项目文件夹中的所有内容打包并作为源包上传到 Elastic Beanstalk。当您结合使用 Git 和 EB CLI 时，可以使用 `.gitignore` 防止将生成的类文件提交到源，并使用 `.ebignore` 防止部署源文件。

您还可以配置 EB CLI 部署生成项目（WAR 或 ZIP 文件）而不是项目文件夹的内容。

安装 AWS CLI

AWS Command Line Interface (AWS CLI) 是AWS服务的统一客户端，为所有公有 API 操作提供命令。这些命令的级别比 EB CLI 提供的命令的级别低，因此使用 AWS CLI 执行操作所需的命令经常会多一些。另一方面，通过 AWS Command Line Interface，您无需在本机上设置存储库就可以使用在您账户中运行的任何应用程序或环境。使用 AWS CLI 可创建脚本来简化或自动执行操作性任务。

有关支持的服务和下载 AWS Command Line Interface 的更多信息，请参阅 AWS Command Line Interface。

使用 Elastic Beanstalk 命令行界面 (EB CLI)

EB CLI 是一个命令行界面 AWS Elastic Beanstalk ，它提供交互式命令，可简化从本地存储库创建、更新和监控环境。将 EB CLI 用作每日开发和测试周期的一部分，以作为 Elastic Beanstalk 控制台的一种替代方法。

> **Note**
>
> 当前版本的 EB CLI 具有一组与 3.0 版之前的版本不同的基本命令。如果您使用的是较旧版本，请参阅迁移到 EB CLI 3 和 CodeCommit以了解迁移信息。

在安装 EB CLI 并配置项目目录后，您可以使用单个命令创建环境：

```
~/my-app$ eb create my-env
```

EB CLI 的源代码是一个开源项目。它驻留在aws/aws-elastic-beanstalk-cli GitHub 存储库中。您可以报告问题，提出建议和提交拉取请求以参与我们的项目。我们非常欢迎您参与。对于仅打算原样使用 EB CLI 的环境，我们建议您使用 EB CLI 安装脚本之一进行安装，如the section called “使用安装脚本安装 EB CLI”中所述。

此前，Elastic Beanstalk 已支持单独的 CLI，后者提供了对调用 Elastic Beanstalk API CLI 的操作的直接访问权限。它已被替换为 AWS CLI，后者提供相同的功能，但适用于所有 AWS 服务的 API。

有了它， AWS CLI 你可以直接访问 Elastic Beanstalk API。 AWS CLI 非常适合编写脚本，但由于需要运行的命令数量和每条命令的参数数量众多，因此在命令行中使用起来却不那么容易。例如，创建环境需要一系列命令：

```
~$ aws elasticbeanstalk check-dns-availability --cname-prefix my-cname
~$ aws elasticbeanstalk create-application-version --application-name my-application --
version-label v1 --source-bundle S3Bucket=DOC-EXAMPLE-BUCKET,S3Key=php-proxy-sample.zip
~$ aws elasticbeanstalk create-environment --cname-prefix my-cname --application-name
 my-app --version-label v1 --environment-name my-env --solution-stack-name "64bit
 Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Passenger Standalone)"
```

有关如何安装 EB CLI、配置存储库和处理环境的信息，请参阅以下主题。

主题

安装 EB CLI

AWS Elastic Beanstalk Command Line Interface (EB CLI) 是一个可用来创建、配置和管理 Elastic Beanstalk 环境的命令行客户端。有关 EB CLI 的更多信息，请参阅EB CLI。

主题

使用安装脚本安装 EB CLI

安装 EB CLI 的最简单和建议的方法是，使用 GitHub 上提供的 EB CLI 安装脚本。可以使用脚本在 Linux、macOS 或 Windows 上安装 EB CLI。这些脚本安装 EB CLI 及其依赖项，包括 Python 和 `pip`。这些脚本还会为 EB CLI 创建虚拟环境。有关安装说明，请参阅 GitHub 上的 `aws/aws-elastic-beanstalk-cli-setup` 存储库。

手动安装 EB CLI

要安装 EB CLI，我们建议使用 EB CLI 安装脚本。如果安装脚本与您的开发环境不兼容，请手动安装 EB CLI。

EB CLI 在 Linux、macOS 和 Windows 上的主要分发方式为 pip。这是一个用于 Python 的程序包管理器，提供了简单的方式来安装、升级和删除 Python 程序包及其相关组件。对于 macOS，您还可以随 Homebrew 获取最新版本的 EB CLI。

兼容性备注

EB CLI 在 Python 中开发，需要 Python 版本 3.11 或更高版本。

我们建议使用 EB CLI 安装脚本来安装 EB CLI 及其依赖项。如果您手动安装 EB CLI，可能会难以管理开发环境中的依赖项冲突。

EB CLI 和 AWS Command Line Interface (AWS CLI) 共同依赖于 botocore Python 包。由于 botocore 中的重大更改，这两种不同版本的 CLI 工具依赖于不同版本的 botocore。

这两个 CLI 的最新版本是兼容的。如果您需要使用早期版本，请参阅下表，了解要使用的兼容版本。

EB CLI 版本	兼容的 AWS CLI 版本
3.14.5 或更早版本	1.16.9 或更早版本
3.14.6 或更高版本	1.16.11 或更高版本

安装 EB CLI

如果您已经有 pip 和支持的 Python 版本，请使用以下步骤安装 EB CLI。

如果您没有 Python 和 pip，则使用适合正在使用的操作系统的过程：

- 在 Linux 上安装 Python、pip 和 EB CLI
- 在 macOS 上安装 EB CLI
- 在 Windows 上安装 Python、pip 和 EB CLI

安装 EB CLI

1. 运行以下命令。

```
$ pip install awsebcli --upgrade --user
```

`--upgrade` 选项通知 `pip` 升级已安装的任何必要组件。`--user` 选项通知 `pip` 将程序安装到用户目录的子目录中，以避免修改您的操作系统所使用的库。

> **Note**
>
> 如果您尝试随 `pip` 一起安装 EB CLI 时遇到问题，可以在虚拟环境中安装 EB CLI 来隔离工具及其依赖对象；或者使用与平时使用的 Python 不同的版本。

2. 将可执行文件的路径添加到您的 PATH 变量中：

   - 在 Linux 和 macOS 上：

     Linux – `~/.local/bin`

     macOS – `~/Library/Python/3.7/bin`

     要修改您的 PATH 变量（Linux、Unix 或 macOS），请执行以下操作：

     a. 在您的用户文件夹中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 `echo $SHELL`。

     ```
     $ ls -a ~
     .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
     ```

     - Bash – `.bash_profile`、`.profile` 或 `.bash_login`。
     - Zsh – `.zshrc`
     - Tcsh – `.tcshrc`、`.cshrc` 或 `.login`。

     b. 向配置文件脚本中添加导出命令。以下示例向当前 PATH 变量中添加 *LOCAL_PATH* 所表示的路径。

     ```
     export PATH=LOCAL_PATH:$PATH
     ```

     c. 将在第一步中描述的配置脚本文件加载到当前会话中。以下示例加载 *PROFILE_SCRIPT* 所表示的配置文件脚本。

     ```
     $ source ~/PROFILE_SCRIPT
     ```

   - 在 Windows 上：

Python 3.7 – `%USERPROFILE%\AppData\Roaming\Python\Python37\Scripts`

Python 早期版本 – `%USERPROFILE%\AppData\Roaming\Python\Scripts`

要修改您的 PATH 变量 (Windows)，请执行以下操作：

a. 按下 Windows 键，然后键入 **environment variables**。
b. 选择 Edit environment variables for your account（编辑您账户的环境变量）。
c. 选择 PATH，然后选择 Edit（编辑）。
d. 向 Variable value 字段添加路径，中间用分号隔开。例如：*C:\item1\path*;*C:\item2\path*
e. 选择 OK 两次以应用新设置。
f. 关闭任何正在运行的命令提示符窗口，然后重新打开命令提示符窗口。

3. 通过运行 eb --version 来验证 EB CLI 是否已正确安装。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

EB CLI 会定期更新以增加支持最新 Elastic Beanstalk 特性的功能。要更新到最新版本的 EB CLI，请再次运行安装命令。

```
$ pip install awsebcli --upgrade --user
```

如果需要卸载 EB CLI，请使用 `pip uninstall`。

```
$ pip uninstall awsebcli
```

在 Linux 上安装 Python、pip 和 EB CLI

EB CLI 需要 Python 2.7、3.4 或更高版本。如果您的分发没有随 Python 提供，或者提供的是较早的版本，请在安装 `pip` 和 EB CLI 之前安装 Python。

在 Linux 上安装 Python 3.7

1. 确定是否已安装 Python。

```
$ python --version
```

> **Note**
>
> 如果您的 Linux 分发版本附带了 Python，则可能需要安装 Python 开发人员程序包以获取编译扩展和安装 EB CLI 时需要的标头和库。使用程序包管理器安装开发人员程序包（名称通常为 `python-dev` 或 `python-devel`）。

2. 如果尚未安装 Python 2.7 或更高版本，请使用分发版本的程序包管理器来安装 Python 3.7。命令和程序包名称会有所不同：

   - 在 Debian 衍生系统（如 Ubuntu）上，请使用 APT：

     ```
     $ sudo apt-get install python3.7
     ```

   - 在 Red Hat 及其衍生系统上，请使用 `yum`。

     ```
     $ sudo yum install python37
     ```

   - 在 SUSE 及其衍生系统上，请使用 `zypper`。

     ```
     $ sudo zypper install python3-3.7
     ```

3. 要验证是否已正确安装 Python，请打开终端或 Shell，并运行以下命令。

   ```
   $ python3 --version
   Python 3.7.3
   ```

使用 Python 打包权威机构提供的脚本安装 `pip`，然后安装 EB CLI。

安装 **`pip`** 和 EB CLI

1. 从 pypa.io 下载安装脚本。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   ```

   脚本将会下载，并将安装 `pip` 的最新版本以及另一个名为 `setuptools` 的必需程序包。

2. 使用 Python 运行脚本。

```
$ python3 get-pip.py --user
Collecting pip
  Downloading pip-8.1.2-py2.py3-none-any.whl (1.2MB)
Collecting setuptools
  Downloading setuptools-26.1.1-py2.py3-none-any.whl (464kB)
Collecting wheel
  Downloading wheel-0.29.0-py2.py3-none-any.whl (66kB)
Installing collected packages: pip, setuptools, wheel
Successfully installed pip setuptools wheel
```

使用 python3 命令而不是 python 来直接调用 Python 版本 3，这样可确保即使系统上存在 Python 的较旧版本，pip 也会安装在正确的位置。

3. 将可执行文件的路径 ~/.local/bin 添加到您的 PATH 变量中：

要修改您的 PATH 变量（Linux、Unix 或 macOS），请执行以下操作：

a. 在您的用户文件夹中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 echo $SHELL。

```
$ ls -a ~
.  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
```

- Bash – .bash_profile、.profile 或 .bash_login。
- Zsh – .zshrc
- Tcsh – .tcshrc、.cshrc 或 .login。

b. 向配置文件脚本中添加导出命令。以下示例向当前 PATH 变量中添加 *LOCAL_PATH* 所表示的路径。

```
export PATH=LOCAL_PATH:$PATH
```

c. 将在第一步中描述的配置脚本文件加载到当前会话中。以下示例加载 *PROFILE_SCRIPT* 所表示的配置文件脚本。

```
$ source ~/PROFILE_SCRIPT
```

4. 验证 pip 是否已正确安装。

```
$ pip --version
pip 8.1.2 from ~/.local/lib/python3.7/site-packages (python 3.7)
```

5. 使用 pip 安装 EB CLI。

```
$ pip install awsebcli --upgrade --user
```

6. 验证 EB CLI 是否已正确安装。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

要升级到最新版本，请重新运行安装命令。

```
$ pip install awsebcli --upgrade --user
```

在 macOS 上安装 EB CLI

如果您使用 Homebrew 程序包管理器，可使用 brew 命令安装 EB CLI。您也可安装 Python 和 pip，然后使用 pip 安装 EB CLI。

使用 Homebrew 安装 EB CLI

当 pip 中显示了最新版本的 EB CLI 时，通常数日之后即在 Homebrew 中可用。

使用 **Homebrew** 安装 EB CLI

1. 确保您有最新版本的 Homebrew。

```
$ brew update
```

2. 运行 brew install awsebcli。

```
$ brew install awsebcli
```

3. 验证 EB CLI 是否已正确安装。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

在 macOS 上安装 Python、pip 和 EB CLI

您可安装 Python 和 pip 的最新版本，然后用其安装 EB CLI。

在 macOS 上安装 EB CLI

1. 从 Python.org 的下载页面下载并安装 Python。我们使用版本 3.7 进行演示。

   > **Note**
   >
   > EB CLI 需要 Python 2 版本 2.7 或者版本范围在 3.4 和 3.7 之间的 Python 3。

2. 使用 Python 打包权威机构提供的脚本来安装 pip。

   ```
   $ curl -O https://bootstrap.pypa.io/get-pip.py
   $ python3 get-pip.py --user
   ```

3. 使用 pip 安装 EB CLI。

   ```
   $ pip3 install awsebcli --upgrade --user
   ```

4. 将可执行文件的路径 ~/Library/Python/3.7/bin 添加到您的 PATH 变量中：

   要修改您的 PATH 变量（Linux、Unix 或 macOS），请执行以下操作：

   a. 在您的用户文件夹中查找 Shell 的配置文件脚本。如果您不能确定所使用的 Shell，请运行 echo $SHELL。

      ```
      $ ls -a ~
      .  ..  .bash_logout  .bash_profile  .bashrc  Desktop  Documents  Downloads
      ```

      - Bash – .bash_profile、.profile 或 .bash_login。
      - Zsh – .zshrc
      - Tcsh – .tcshrc、.cshrc 或 .login。

   b. 向配置文件脚本中添加导出命令。以下示例向当前 PATH 变量中添加 *LOCAL_PATH* 所表示的路径。

      ```
      export PATH=LOCAL_PATH:$PATH
      ```

c. 将在第一步中描述的配置脚本文件加载到当前会话中。以下示例加载 *PROFILE_SCRIPT* 所表示的配置文件脚本。

```
$ source ~/PROFILE_SCRIPT
```

5. 验证 EB CLI 是否已正确安装。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

要升级到最新版本，请重新运行安装命令。

```
$ pip3 install awsebcli --upgrade --user
```

在 Windows 上安装 Python、pip 和 EB CLI

Python Software Foundation 为包含 `pip` 的 Windows 提供了安装程序。

安装 Python 3.6 和 **`pip`** (Windows)

1. 从 Python.org 的下载页面下载最新的 Python Windows x86-64 可执行文件安装程序。
2. 运行在上一步中下载的 Python 安装程序可执行文件。

   从 Python 安装程序窗口中选择以下选项，以为接下来的 EB CLI 安装步骤进行设置。

   a. 选择将 Python 可执行文件添加到您的路径。

   b. 选择 Install Now。

> **Note**
>
> 有关安装选项的更多信息，请参阅Python 网站上的在 Windows 上使用 Python 页面。文档网站在页面顶部提供了一个下拉列表，您可以在其中选择文档的 Python 版本。

安装程序在您的用户文件夹中安装 Python 并将其可执行文件目录添加到您的用户路径。

使用 **pip** 安装 AWS CLI (Windows)

1. 从“开始”菜单中打开命令提示符窗口。
2. 使用以下命令验证 Python 和 pip 是否已正确安装。

```
C:\Users\myname> python --version
Python 3.11.4
C:\Users\myname> pip --version
pip 23.1.2 from C:\Users\myname\AppData\Local\Programs\Python\Python311\Lib\site-
packages\pip (python 3.11)
```

3. 使用 pip 安装 EB CLI。

```
C:\Users\myname> pip install awsebcli --upgrade --user
```

4. 将以下可执行文件路径添加到 Windows 用户帐户中的 Path 环境变量。安装 Python 的位置可能有所不同，具体取决于您为一个用户还是为所有用户安装。

   %USERPROFILE%\AppData\Roaming\Python\Python311\Scripts

5. 重新启动新的命令 shell，以使新的 Path 变量生效。
6. 验证 EB CLI 是否已正确安装。

```
C:\Users\myname>  eb --version
EB CLI 3.14.8 (Python 3.11)
```

要升级到最新版本，请重新运行安装命令。

```
C:\Users\myname> pip install awsebcli --upgrade --user
```

在虚拟环境中安装 EB CLI

您可以通过在虚拟环境中安装 EB CLI 来避免所需版本与其他 pip 程序包发生冲突。

在虚拟环境中安装 EB CLI

1. 使用 pip 安装 virtualenv。

```
$ pip install --user virtualenv
```

2. 创建虚拟环境。

```
$ virtualenv ~/eb-ve
```

要使用默认可执行文件之外的其他 Python 可执行文件，请使用 -p 选项。

```
$ virtualenv -p /usr/bin/python3.7 ~/eb-ve
```

3. 激活虚拟环境。

Linux、Unix 或 macOS

```
$ source ~/eb-ve/bin/activate
```

Windows

```
$ %USERPROFILE%\eb-ve\Scripts\activate
```

4. 安装 EB CLI。

```
(eb-ve)~$ pip install awsebcli --upgrade
```

5. 验证 EB CLI 是否已正确安装。

```
$ eb --version
EB CLI 3.14.8 (Python 3.7)
```

您可以使用 `deactivate` 命令退出虚拟环境。只要启动了新的会话，请重新运行激活命令。

要升级到最新版本，请重新运行安装命令。

```
(eb-ve)~$ pip install awsebcli --upgrade
```

配置 EB CLI

在安装 EB CLI 后，就可以运行 eb init 来配置项目目录和 EB CLI 了。

下面的示例说明在名为 eb init 的项目文件夹中首次运行 eb 时的配置步骤。

初始化 EB CLI 项目

1. 首先，EB CLI 会提示您选择一个区域。键入与要使用的区域对应的编号，然后按 Enter。

```
~/eb $ eb init
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 3
```

2. 然后，提供您的访问密钥和私有密钥以便 EB CLI 可以管理资源。访问密钥是在 AWS Identity and Access Management 控制台中创建的。如果您没有密钥，请参阅 Amazon Web Services 一般参考 中的如何获取安全凭证？。

```
You have not yet set up your credentials or your credentials are incorrect.
You must provide your credentials.
(aws-access-id): AKIAJOUAASEXAMPLE
(aws-secret-key): 5ZRIrtTM4ciIAvd4EXAMPLEDtm+PiPSzpoK
```

3. Elastic Beanstalk 中的应用程序是一项资源，它包含一组与单个 Web 应用程序关联的应用程序版本（源）、环境和保存的配置。每次您使用 EB CLI 将源代码部署到 Elastic Beanstalk 时，都会创建一个新的应用程序版本并将其添加到列表中。

```
Select an application to use
1) [ Create new Application ]
(default is 1): 1
```

4. 默认应用程序名是您在其中运行 eb init 的文件夹的名称。输入可描述项目的任意名称。

```
Enter Application Name
(default is "eb"): eb
Application eb has been created.
```

5. 选择与您的 Web 应用程序的开发语言或框架相符的平台。如果您尚未开始开发应用程序，请选择您喜欢的平台。您将很快了解如何启动示例应用程序，并且以后随时可以更改该设置。

```
Select a platform.
1) Node.js
2) PHP
3) Python
4) Ruby
5) Tomcat
6) IIS
7) Docker
8) Multi-container Docker
9) GlassFish
10) Go
11) Java
(default is 1): 1
```

6. 选择 yes (是) 以将 SSH 密钥对分配给 Elastic Beanstalk 环境中的实例。这允许您直接连接到它们以进行故障排除。

```
Do you want to set up SSH for your instances?
(y/n): y
```

7. 选择现有密钥对或创建新密钥对。要使用 eb init 创建新的密钥对，ssh-keygen 必须已在本地计算机上安装且能够从命令行访问。EB CLI 向 Amazon EC2 注册新的密钥对，并将私有密钥本地存储在用户目录中名为 `.ssh` 的文件夹中。

```
Select a keypair.
1) [ Create new KeyPair ]
(default is 1): 1
```

您的 EB CLI 安装现已配置，可以使用。有关创建和使用 Elastic Beanstalk 环境的说明，请参阅使用 EB CLI 管理 Elastic Beanstalk 环境。

高级配置

- 使用 .ebignore 忽略文件
- 使用命名配置文件
- 部署构件而不是项目文件夹
- 配置设置和优先顺序
- 实例元数据

使用 .ebignore 忽略文件

您可将 .ebignore 文件添加到目录中，告知 EB CLI 忽略您的项目目录中的某些文件。此文件的作用与 .gitignore 类似。当您将项目目录部署到 Elastic Beanstalk 并创建新的应用程序版本时，EB CLI 不会将 .ebignore 指定的文件包含在其创建的源包中。

如果 .ebignore 不存在，但存在 .gitignore，EB CLI 将忽略 .gitignore 中指定的文件。如果存在 .ebignore，EB CLI 将不会读取 .gitignore。

如果 .ebignore 存在，EB CLI 不会使用 git 命令创建您的源包。这就意味着 EB CLI 将忽略 .ebignore 中指定的文件，并包括所有其他文件。具体而言，它会包括未提交的源文件。

> **Note**
>
> 在 Windows 中，在创建源包时添加 .ebignore 将导致 EB CLI 访问符号链接并包含链接的文件。这是一个已知问题，将在未来的更新中得到修复。

使用命名配置文件

如果将凭证作为命名配置文件存储在 credentials 或 config 文件中，则可使用 --profile 选项显式指定配置文件。例如，以下命令将使用 user2 配置文件创建新应用程序。

```
$ eb init --profile user2
```

您也可以通过设置 AWS_EB_PROFILE 环境变量更改默认配置文件。设置此变量后，EB CLI 从指定的配置文件而不是 default 或 eb-cli 读取凭证。

Linux、macOS 或 Unix

```
$ export AWS_EB_PROFILE=user2
```

Windows

```
> set AWS_EB_PROFILE=user2
```

部署构件而不是项目文件夹

通过将以下行添加到项目文件夹中的 .elasticbeanstalk/config.yml，可以告诉 EB CLI 部署在单独构建过程中生成的 ZIP 文件或 WAR 文件。

```
deploy:
  artifact: path/to/buildartifact.zip
```

如果您在 Git 存储库中配置了 EB CLI，但未将工件提交到源，请使用 `--staged` 选项部署最新内部版本。

```
~/eb$ eb deploy --staged
```

配置设置和优先顺序

EB CLI 使用提供程序链在很多不同位置查找AWS凭证，包括系统或用户环境变量以及本地AWS配置文件。

EB CLI 按以下顺序查找凭证和配置设置：

1. 命令行选项 - 使用 `--profile` 指定命名配置文件来覆盖默认设置。
2. 环境变量 – `AWS_ACCESS_KEY_ID` 和 `AWS_SECRET_ACCESS_KEY`。
3. AWS凭证文件 – 在 Linux 和 OS X 系统上位于 `~/.aws/credentials`，在 Windows 系统上位于 `C:\Users\USERNAME\.aws\credentials`。除默认配置文件外，此文件还可包含多个命名配置文件。
4. AWS CLI 配置文件 – 在 Linux 和 OS X 系统上位于 `~/.aws/config`，在 Windows 系统上位于 `C:\Users\USERNAME\.aws\config`。此文件可以包含默认配置文件、命名配置文件及各自的 AWS CLI 特定配置参数。
5. 传统 EB CLI 配置文件 – 在 Linux 和 OS X 系统上位于 `~/.elasticbeanstalk/config`，在 Windows 系统上位于 `C:\Users\USERNAME\.elasticbeanstalk\config`。
6. 实例配置文件凭证 – 这些凭证可以通过指定的实例角色用于 Amazon EC2 实例，并通过 Amazon EC2 元数据服务提供。实例配置文件必须有权使用 Elastic Beanstalk。

如果凭证文件包含名为“eb-cli”的命名配置文件，则 EB CLI 优先选择该配置文件而不是默认配置文件。如果未找到配置文件或找到配置文件但无权使用 Elastic Beanstalk，则 EB CLI 会提示您输入密钥。

实例元数据

要从 Amazon EC2 实例使用 EB CLI，请创建一个对所需资源有访问权限的角色，然后在实例启动时将该角色分配给实例。启动实例并使用 `pip` 安装 EB CLI。

```
~$ sudo pip install awsebcli
```

pip 预安装在 Amazon Linux 上。

EB CLI 将从实例元数据读取凭证。有关更多信息，请参阅 IAM 用户指南中的向 Amazon EC2 实例中运行的应用程序授予对AWS资源的访问权限。

使用 EB CLI 管理 Elastic Beanstalk 环境

在安装 EB CLI 和配置项目目录之后，您便可使用 EB CLI 创建 Elastic Beanstalk 环境、部署源和配置更新以及拉取日志和事件。

> **Note**
>
> 使用 EB CLI 创建环境需要服务角色。您可以通过在 Elastic Beanstalk 控制台中创建环境来创建服务角色。如果您没有服务角色，EB CLI 会尝试在您运行 `eb create` 时创建一个。

EB CLI 将对所有成功的命令返回零 (0) 退出代码，并在遇到任何错误时返回非零退出代码。

下面的示例使用一个名为 eb 的空项目文件夹，该文件夹是用 EB CLI 初始化来供示例 Docker 应用程序使用的。

基本命令

- Eb create
- Eb status
- Eb health
- Eb events
- Eb logs
- Eb open
- Eb deploy
- Eb config
- Eb terminate

Eb create

要创建您的第一个环境，请运行 eb create 并按照提示操作。如果项目目录中有源代码，EB CLI 会将源代码捆绑并部署到环境。否则将使用示例应用程序。

```
~/eb$ eb create
Enter Environment Name
(default is eb-dev): eb-dev
Enter DNS CNAME prefix
(default is eb-dev): eb-dev
WARNING: The current directory does not contain any source code. Elastic Beanstalk is
 launching the sample application instead.
Environment details for: elasticBeanstalkExa-env
  Application name: elastic-beanstalk-example
  Region: us-west-2
  Deployed Version: Sample Application
  Environment ID: e-j3pmc8tscn
  Platform: 64bit Amazon Linux 2015.03 v1.4.3 running Docker 1.6.2
  Tier: WebServer-Standard
  CNAME: eb-dev.elasticbeanstalk.com
  Updated: 2015-06-27 01:02:24.813000+00:00
Printing Status:
INFO: createEnvironment is starting.
 -- Events -- (safe to Ctrl+C) Use "eb abort" to cancel the command.
```

环境可能需要几分钟才能就绪。创建环境后，按 Ctrl+C 可返回命令行。

Eb status

运行 eb status 可查看环境的当前状态。如果状态为 `ready`，则示例应用程序在 elasticbeanstalk.com 可用，并且环境可以更新。

```
~/eb$ eb status
Environment details for: elasticBeanstalkExa-env
  Application name: elastic-beanstalk-example
  Region: us-west-2
  Deployed Version: Sample Application
  Environment ID: e-gbzqc3jcra
  Platform: 64bit Amazon Linux 2015.03 v1.4.3 running Docker 1.6.2
  Tier: WebServer-Standard
  CNAME: elasticbeanstalkexa-env.elasticbeanstalk.com
  Updated: 2015-06-30 01:47:45.589000+00:00
  Status: Ready
  Health: Green
```

Eb health

使用 eb health 命令查看有关您环境中的实例的运行状况信息以及环境的整体状态。使用 --refresh 选项可在一个每 10 秒更新一次的交互式视图中查看运行状况。

```
~/eb$ eb health
 api                                  Ok                2016-09-15 18:39:04
WebServer                                                          Java 8
  total      ok    warning  degraded  severe    info   pending  unknown
    3        3        0        0        0        0        0        0

  instance-id           status     cause                              health
    Overall             Ok
  i-0ef05ec54918bf567   Ok
  i-001880c1187493460   Ok
  i-04703409d90d7c353   Ok

  instance-id           r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75
   p50     p10
    Overall             8.6     100.0    0.0    0.0    0.0    0.083*   0.065    0.053
 0.040   0.019
  i-0ef05ec54918bf567   2.9        29      0      0      0    0.069*   0.066    0.057
 0.050   0.023
  i-001880c1187493460   2.9        29      0      0      0    0.087*   0.069    0.056
 0.050   0.034
  i-04703409d90d7c353   2.8        28      0      0      0    0.051*   0.027    0.024
 0.021   0.015

  instance-id           type       az   running     load 1  load 5      user%  nice%
 system%  idle%   iowait%
  i-0ef05ec54918bf567   t2.micro   1c   23 mins       0.19    0.05        3.0    0.0
   0.3   96.7       0.0
  i-001880c1187493460   t2.micro   1a   23 mins        0.0     0.0        3.2    0.0
   0.3   96.5       0.0
  i-04703409d90d7c353   t2.micro   1b   1 day          0.0     0.0        3.6    0.0
   0.2   96.2       0.0

  instance-id           status     id   version                 ago
            deployments
  i-0ef05ec54918bf567   Deployed   28   app-bc1b-160915_181041   20 mins
  i-001880c1187493460   Deployed   28   app-bc1b-160915_181041   20 mins
  i-04703409d90d7c353   Deployed   28   app-bc1b-160915_181041   27 mins
```

Eb events

使用 eb events 可查看 Elastic Beanstalk 输出的事件列表。

```
~/eb$ eb events
2015-06-29 23:21:09    INFO    createEnvironment is starting.
2015-06-29 23:21:10    INFO    Using elasticbeanstalk-us-east-2-EXAMPLE as Amazon S3
 storage bucket for environment data.
2015-06-29 23:21:23    INFO    Created load balancer named: awseb-e-g-AWSEBLoa-EXAMPLE
2015-06-29 23:21:42    INFO    Created security group named: awseb-e-gbzqc3jcra-stack-
AWSEBSecurityGroup-EXAMPLE
...
```

Eb logs

使用 eb logs 可从环境中的实例推送日志。默认情况下，eb logs 从启动的第一个实例推送日志并将其显示在标准输出中。通过 --instance 选项指定实例 ID 可从特定实例获取日志。

--all 选项从所有实例提取日志并将它们保存到 `.elasticbeanstalk/logs` 下的子目录中。

```
~/eb$ eb logs --all
Retrieving logs...
Logs were saved to /home/local/ANT/mwunderl/ebcli/environments/test/.elasticbeanstalk/
logs/150630_201410
Updated symlink at /home/local/ANT/mwunderl/ebcli/environments/test/.elasticbeanstalk/
logs/latest
```

Eb open

要在浏览器中打开环境的网站，请使用 eb open：

```
~/eb$ eb open
```

在窗口式环境中，默认浏览器将在新窗口中打开。在终端环境中，如果可用，将使用命令行浏览器 (例如 w3m)。

Eb deploy

环境启动并就绪后，可以使用 eb deploy 更新环境。

此命令与一些源代码使用可以更好地捆绑和部署，对于此示例，我们已在项目目录中使用以下内容创建了一个 Dockerfile：

~/eb/Dockerfile

```
FROM ubuntu:12.04

RUN apt-get update
RUN apt-get install -y nginx zip curl

RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN curl -o /usr/share/nginx/www/master.zip -L https://codeload.github.com/
gabrielecirulli/2048/zip/master
RUN cd /usr/share/nginx/www/ && unzip master.zip && mv 2048-master/* . && rm -rf 2048-
master master.zip

EXPOSE 80

CMD ["/usr/sbin/nginx", "-c", "/etc/nginx/nginx.conf"]
```

此 Dockerfile 部署一个 Ubuntu 12.04 映像并安装游戏 2048。运行 eb deploy 可将该应用程序上传到您的环境中：

```
~/eb$ eb deploy
Creating application version archive "app-150630_014338".
Uploading elastic-beanstalk-example/app-150630_014338.zip to S3. This may take a while.
Upload Complete.
INFO: Environment update is starting.
 -- Events -- (safe to Ctrl+C) Use "eb abort" to cancel the command.
```

运行 eb deploy 时，EB CLI 会捆绑项目目录中的内容并将其部署到环境中。

> **Note**
>
> 如果您已在项目文件夹中初始化了一个 Git 存储库，EB CLI 将始终部署最新提交，即使您有挂起的更改。在运行 eb deploy 前提交更改可将它们部署到环境中。

Eb config

使用 eb config 命令可查看您的运行环境可用的配置选项：

```
~/eb$ eb config
ApplicationName: elastic-beanstalk-example
DateUpdated: 2015-06-30 02:12:03+00:00
EnvironmentName: elasticBeanstalkExa-env
SolutionStackName: 64bit Amazon Linux 2015.03 v1.4.3 running Docker 1.6.2
settings:
  AWSEBAutoScalingScaleDownPolicy.aws:autoscaling:trigger:
    LowerBreachScaleIncrement: '-1'
  AWSEBAutoScalingScaleUpPolicy.aws:autoscaling:trigger:
    UpperBreachScaleIncrement: '1'
  AWSEBCloudwatchAlarmHigh.aws:autoscaling:trigger:
    UpperThreshold: '6000000'
...
```

此命令将可用配置选项列表填充在文本编辑器中。所示的很多选项带有 null 值，这不是默认设置，可以进行修改以更新环境中的资源。有关这些选项的更多信息，请参阅配置选项。

Eb terminate

如果现在环境使用完毕，可使用 eb terminate 将其终止。

```
~/eb$ eb terminate
The environment "eb-dev" and all associated instances will be terminated.
To confirm, type the environment name: eb-dev
INFO: terminateEnvironment is starting.
INFO: Deleted CloudWatch alarm named: awseb-e-jc8t3pmscn-stack-
AWSEBCloudwatchAlarmHigh-1XLMU7DNCBV6Y
INFO: Deleted CloudWatch alarm named: awseb-e-jc8t3pmscn-stack-
AWSEBCloudwatchAlarmLow-8IVI04W2SCXS
INFO: Deleted Auto Scaling group policy named: arn:aws:autoscaling:us-
east-2:123456789012:scalingPolicy:1753d43e-ae87-4df6-
a405-11d31f4c8f97:autoScalingGroupName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingGroup-90TTS2ZL4MXV:policyName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingScaleUpPolicy-A070H1BMUQAJ
INFO: Deleted Auto Scaling group policy named: arn:aws:autoscaling:us-
east-2:123456789012:scalingPolicy:1fd24ea4-3d6f-4373-
affc-4912012092ba:autoScalingGroupName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingGroup-90TTS2ZL4MXV:policyName/awseb-e-jc8t3pmscn-stack-
AWSEBAutoScalingScaleDownPolicy-LSWFUMZ46H1V
INFO: Waiting for EC2 instances to terminate. This may take a few minutes.
 -- Events -- (safe to Ctrl+C)
```

有关可用 EB CLI 命令的完整列表，请参阅 EB CLI 命令参考。

> **Important**
>
> 如果您终止环境，则还必须删除您创建的任何 CNAME 映射，因为其他客户可能会重用可用的主机名。请务必删除指向已终止环境的 DNS 记录，以防出现 悬空 DNS 条目。悬空 DNS 条目可能会使指向您的域的互联网流量出现安全漏洞，此外还可能带来其他风险。
> 有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的 防止 Route 53 中悬挂委派记录。您还可以访问《AWS 安全博客》中的 针对 Amazon CloudFront 请求的增强域保护，了解有关悬空 DNS 条目的更多信息。

通过 AWS CodeBuild 使用 EB CLI

AWS CodeBuild 可编译源代码，运行单元测试，并生成可供部署的项目。您可以将 CodeBuild 与 EB CLI 结合使用来从应用程序的源代码自动构建应用程序。环境创建和之后的每个部署都将以构建步骤开始，然后部署生成的应用程序。

> **Note**
>
> 有些区域不提供 CodeBuild。Elastic Beanstalk 和 CodeBuild 之间的集成在这些区域无法正常工作。
> 有关每个区域中提供的 AWS 服务，请参阅区域表。

创建应用程序

创建使用 CodeBuild 的 Elastic Beanstalk 应用程序

1. 在您的应用程序文件夹中包含 CodeBuild 构建规范文件 `buildspec.yml`。
2. 使用特定于 Elastic Beanstalk 的选项添加一个 `eb_codebuild_settings` 条目到文件。
3. 在此文件夹中运行 eb init。

> **Note**
>
> 将 EB CLI 与 CodeBuild 一起使用时，请勿在 Application name（应用程序名称）中使用句点 (.) 或空格 ( ) 字符。

Elastic Beanstalk 扩展 CodeBuild 构建规范文件格式，以包括以下各项其他设置：

```
eb_codebuild_settings:
  CodeBuildServiceRole: role-name
  ComputeType: size
  Image: image
  Timeout: minutes
```

`CodeBuildServiceRole`

CodeBuild 可用来代表您与相关AWS服务进行交互的 AWS Identity and Access Management (IAM) 服务角色的 ARN 或名称。此值为必填项。如果您忽略此值，任何后续的 eb create 或 eb deploy 命令都将失败。

要了解有关为 CodeBuild 创建服务角色的更多信息，请参阅《AWS CodeBuild 用户指南》中的创建 CodeBuild 服务角色。

> **Note**
>
> 您还需要在 CodeBuild 本身中执行操作的权限。Elastic Beanstalk AdministratorAccess-AWSElasticBeanstalk 托管用户策略包含所有必需的 CodeBuild 操作权限。如果您不使用托管策略，请确保在用户策略中允许以下权限。
>
> ```
> "codebuild:CreateProject",
> "codebuild:DeleteProject",
> "codebuild:BatchGetBuilds",
> "codebuild:StartBuild"
> ```
>
> 有关详细信息，请参阅 管理 Elastic Beanstalk 用户策略。

`ComputeType`

CodeBuild 构建环境中的 Docker 容器使用的资源量。有效值为 BUILD_GENERAL1_SMALL、BUILD_GENERAL1_MEDIUM 和 BUILD_GENERAL1_LARGE。

`Image`

CodeBuild 用于构建环境的 Docker Hub 或 Amazon ECR 映像的名称。此 Docker 映像应包含构建代码所需的所有工具和运行时库，并且应与您应用程序的目标平台匹配。CodeBuild 管理和维护

了一组专门用于 Elastic Beanstalk 的映像。建议您使用其中一个。有关详细信息，请参阅《AWS CodeBuild 用户指南》中的 CodeBuild 提供的 Docker 映像。

`Image` 值是可选的。如果您省略此值，eb init 命令会尝试选择与目标平台最匹配的映像。此外，如果您在交互模式下运行 eb init 并且它无法为您选择映像，则会提示您选择一个映像。成功结束初始化时，eb init 将选中的映像写入到 `buildspec.yml` 文件中。

`Timeout`

CodeBuild 构建的版本在超时之前运行的持续时间，以分钟为单位。该值为可选项。有关有效值和默认值的详细信息，请参阅在 CodeBuild 中创建构建项目。

> **Note**
>
> 此超时控制 CodeBuild 运行的最大持续时间，EB CLI 也会将它作为其创建应用程序版本的第一步的组成部分。它与您通过 eb create 或 eb deploy 命令的 `--timeout` 选项指定的值截然不同。后一个值控制 EB CLI 等待环境创建或更新的最大持续时间。

构建和部署您的应用程序代码

只要您的应用程序代码需要进行部署，EB CLI 就会使用 CodeBuild 运行构建任务，然后将生成的构建构件部署到您的环境。当您使用 eb create 命令为应用程序创建 Elastic Beanstalk 环境时，以及您以后每次使用 eb deploy 命令将代码更改部署到环境时，就会发生上述这种情况。

如果 CodeBuild 步骤失败，则环境创建或部署不启动。

将 EB CLI 与 Git 配合使用

EB CLI 提供了与 Git 的集成。本部分概述如何将 Git 与 EB CLI 配合使用。

安装 Git 并初始化 Git 存储库

1. 转至 http://git-scm.com 下载最新版的 Git。
2. 通过键入以下命令初始化您的 Git 存储库：

```
~/eb$ git init
```

EB CLI 现在将识别出您的应用程序已设置 Git。

3. 如果您尚未运行 eb init，请立即运行：

```
~/eb$ eb init
```

将 Elastic Beanstalk 环境与 Git 分支关联

您可以将不同的环境与各个代码分支关联。在签出分支时，更改将部署到关联的环境。例如，您可以键入以下内容以将生产环境与主线分支关联，并将单独的开发环境与开发分支关联：

```
~/eb$ git checkout mainline
~/eb$ eb use prod
~/eb$ git checkout develop
~/eb$ eb use dev
```

部署更改

默认情况下，EB CLI 将在当前分支中部署最新提交，使用提交 ID 和消息分别作为应用程序版本标签和描述。如果要在不提交的情况下部署到环境，可使用 `--staged` 选项来部署已添加到暂存区域的更改。

在不提交的情况下部署更改

1. 将新文件和更改后的文件添加到暂存区域：

```
~/eb$ git add .
```

2. 使用 eb deploy 部署暂存的更改：

```
~/eb$ eb deploy --staged
```

如果您已配置 EB CLI 以部署工件，并且未将工件提交到 Git 存储库，则使用 `--staged` 选项来部署最新版本。

使用 Git 子模块

通过包含 Git 子模块获益的一些代码项目 - 顶级存储库中的存储库。在使用 eb create 或 eb deploy 部署代码时，EB CLI 可在应用程序版本 zip 文件中包含子模块，并将这些子模块与剩余代码一起上传。

您可以使用项目文件夹中的 EB CLI 配置文件 `include_git_submodules` 的 `global` 部分中的 `.elasticbeanstalk/config.yml` 选项来控制子模块的包含。

要包含子模块，请将此选项设置为 `true`：

```
global:
  include_git_submodules: true
```

当 `include_git_submodules` 选项缺失或设置为 `false` 时，EB CLI 不会在上传的 zip 文件中包含子模块。

有关 Git 子模块的更多详细信息，请参阅 Git 工具 - 子模块。

> **默认行为**
>
> 当您运行 eb init 以配置项目时，EB CLI 会添加 `include_git_submodules` 选项，并将其设置为 `true`。这将确保您的部署中包含项目中已有的所有子模块。
> EB CLI 并不总是支持包含子模块。为了避免对添加子模块支持之前已存在的项目进行意外的不必要更改，EB CLI 在缺少 `include_git_submodules` 选项时不包含子模块。如果您拥有这些现有项目之一，并且希望在部署中包含子模块，请添加此选项并将其设置为 `true`，如本部分中所述。

> **CodeCommit 行为**
>
> Elastic Beanstalk 与 CodeCommit 的集成目前不支持子模块。如果您允许环境与 CodeCommit 集成，子模块将不会包含在部署中。

将 Git 标签分配给您的应用程序版本

您可使用 Git 标签作为您的版本标签来标识正在您的环境中运行的应用程序版本。例如，请键入以下命令：

```
~/eb$ git tag -a v1.0 -m "My version 1.0"
```

通过 AWS CodeCommit 使用 EB CLI

您可以使用 EB CLI 直接从 AWS CodeCommit 存储库部署应用程序。利用 CodeCommit，部署时您只能上载对存储库的更改，而不是上载整个项目。如果您的项目比较大或 Internet 连接带宽有限，这能为您节省时间和带宽。使用 eb appversion、eb create 或 eb deploy 时，EB CLI 将推送本地提交内容并使用它们创建应用程序版本。

若要部署更改，CodeCommit 集成要求您先提交更改。不过，在开发或调试时，建议您不要推送正在处理的未确认更改。您可以通过暂存您的更改并使用 eb deploy --staged (该命令将执行标准部署) 来避免提交这些更改。或者，您可以先将更改提交到开发或测试分支，仅当代码就绪时再合并到主线分支。利用 eb use，您可以将 EB CLI 配置为从开发分支部署到一个环境并从主线分支部署到其他环境。

> **Note**
>
> 有些区域不提供 CodeCommit。Elastic Beanstalk 和 CodeCommit 之间的集成在这些区域无法正常工作。
> 有关每个区域中提供的AWS服务的信息，请参阅区域表。

小节目录

先决条件

要将 CodeCommit 与 AWS Elastic Beanstalk 配合使用，您需要一个至少包含一次提交的本地 Git 存储库（已有的或新建的）、使用 CodeCommit 的权限以及 CodeCommit 支持的区域中的 Elastic Beanstalk 环境。您的环境和存储库必须位于同一个区域中。

初始化 Git 存储库

1. 运行项目文件夹中的 `git init`。

```
~/my-app$ git init
```

2. 使用 `git add` 暂存您的项目文件。

```
~/my-app$ git add .
```

3. 使用 `git commit` 提交更改。

```
~/my-app$ git commit -m "Elastic Beanstalk application"
```

使用 EB CLI 创建 CodeCommit 存储库

要开始使用 CodeCommit，请运行 eb init。在存储库配置期间，EB CLI 会提示您使用 CodeCommit 存储代码并加快部署。即使您之前已使用 eb init 配置项目，也可以重新运行它来配置 CodeCommit。

使用 EB CLI 创建 CodeCommit 存储库

1. 运行项目文件夹中的 eb init。在配置期间，EB CLI 会询问您是否要使用 CodeCommit 存储代码并加快部署。如果您之前已使用 eb init 配置项目，则仍可以重新运行它来配置 CodeCommit。在提示符处键入 **y** 以设置 CodeCommit。

```
~/my-app$ eb init
Note: Elastic Beanstalk now supports AWS CodeCommit; a fully-managed source control
 service. To learn more, see Docs: https://aws.amazon.com/codecommit/
Do you wish to continue with CodeCommit? (y/n)(default is n): y
```

2. 选择 Create new Repository (创建新存储库)。

```
Select a repository
1) my-repo
2) [ Create new Repository ]
(default is 2): 2
```

3. 键入存储库名称或按 Enter 接受默认名称。

```
Enter Repository Name
(default is "codecommit-origin"): my-app
Successfully created repository: my-app
```

4. 为您提交的内容选择现有分支，或使用 EB CLI 创建新的分支。

```
Enter Branch Name
```

```
***** Must have at least one commit to create a new branch with CodeCommit *****
(default is "mainline"): ENTER
Successfully created branch: mainline
```

从 CodeCommit 存储库进行部署

在使用 EB CLI 存储库配置 CodeCommit 时，EB CLI 会使用存储库的内容创建源包。在运行 eb deploy 或 eb create 时，EB CLI 会推送新的提交内容，并使用分支的 HEAD 修订来创建将部署到环境中的 EC2 实例的存档。

将 CodeCommit 集成与 EB CLI 配合使用

1. 使用 eb create 创建新环境。

   ```
   ~/my-app$ eb create my-app-env
   Starting environment deployment via CodeCommit
   --- Waiting for application versions to be pre-processed ---
   Finished processing application version app-ac1ea-161010_201918
   Setting up default branch
   Environment details for: my-app-env
     Application name: my-app
     Region: us-east-2
     Deployed Version: app-ac1ea-161010_201918
     Environment ID: e-pm5mvvkfnd
     Platform: 64bit Amazon Linux 2016.03 v2.1.6 running Java 8
     Tier: WebServer-Standard
     CNAME: UNKNOWN
     Updated: 2016-10-10 20:20:29.725000+00:00
   Printing Status:
   INFO: createEnvironment is starting.
   ...
   ```

   EB CLI 使用所跟踪分支中的最新提交内容来创建部署到环境中的应用程序版本。

2. 有新的本地提交内容时，请使用 eb deploy 推送这些提交内容并部署到环境中。

   ```
   ~/my-app$ eb deploy
   Starting environment deployment via CodeCommit
   INFO: Environment update is starting.
   INFO: Deploying new version to instance(s).
   INFO: New application version was deployed to running EC2 instances.
   ```

```
INFO: Environment update completed successfully.
```

3. 若要在提交更改之前测试更改，请使用 `--staged` 选项部署已使用 `git add` 添加到暂存区域的更改。

```
~/my-app$ git add new-file
~/my-app$ eb deploy --staged
```

使用 `--staged` 选项进行部署时，将绕过 CodeCommit 执行标准部署。

配置其他分支和环境

CodeCommit 配置适用于单个分支。您可以使用 eb use 和 eb codesource 配置其他分支或修改当前分支的配置。

配置 CodeCommit 与 EB CLI 的集成

1. 若要更改远程分支，请使用 eb use 命令的 `--source` 选项。

```
~/my-app$ eb use test-env --source my-app/test
```

2. 若要创建新的分支和环境，请检查新的分支，将它推送到 CodeCommit 并创建环境，然后使用 eb use 连接本地分支、远程分支和环境。

```
~/my-app$ git checkout -b production
~/my-app$ git push --set-upstream production
~/my-app$ eb create production-env
~/my-app$ eb use --source my-app/production production-env
```

3. 要以交互方式配置 CodeCommit，请使用 eb codesource codecommit。

```
~/my-app$ eb codesource codecommit
Current CodeCommit setup:
  Repository: my-app
  Branch: test
Do you wish to continue (y/n): y

Select a repository
1) my-repo
2) my-app
3) [ Create new Repository ]
```

```
(default is 2): 2

Select a branch
1) mainline
2) test
3) [ Create new Branch with local HEAD ]
(default is 1): 1
```

4. 要禁用 CodeCommit 集成，请使用 eb codesource local。

```
~/my-app$ eb codesource local
Current CodeCommit setup:
  Repository: my-app
  Branch: mainline
Default set to use local sources
```

使用现有的 CodeCommit 存储库

如果您已经有 CodeCommit 存储库，并且想将其与 Elastic Beanstalk 配合使用，请在本地 Git 存储库的根目录下运行 eb init。

将现有 CodeCommit 存储库与 EB CLI 配合使用

1. 克隆 CodeCommit 存储库。

```
~$ git clone ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-app
```

2. 查看并推送要用于 Elastic Beanstalk 环境的分支。

```
~/my-app$ git checkout -b dev-env
~/my-app$ git push --set-upstream origin dev-env
```

3. 运行 eb init。选择您当前使用的区域、存储库和分支名称。

```
~/my-app$ eb init
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
```

```
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 1
...
Note: Elastic Beanstalk now supports AWS CodeCommit; a fully-managed source control
 service. To learn more, see Docs: https://aws.amazon.com/codecommit/
Do you wish to continue with CodeCommit? (y/n)(default is n): y

Select a repository
1) my-app
2) [ Create new Repository ]
(default is 1): 1

Select a branch
1) mainline
2) dev-env
3) [ Create new Branch with local HEAD ]
(default is 2): 2
```

有关使用 eb init 的更多信息，请参阅配置 EB CLI。

使用 EB CLI 监控环境的运行状况

Elastic Beanstalk 命令行界面 (EB CLI) 是用于管理 AWS Elastic Beanstalk 环境的命令行工具。您也可以使用 EB CLI 实时监控您的环境的运行状况，这比当前在 Elastic Beanstalk 控制台中提供的监控效果更精细。

在安装并配置 EB CLI 后，您可以使用 eb create 命令启动新环境并将您的代码部署到该环境。如果您已具有在 Elastic Beanstalk 控制台中创建的环境，则可以在项目文件夹中运行 eb init 并按提示操作来将 EB CLI 附加到该环境（项目文件夹可能是空的）。

Important

通过运行带 `pip install` 选项的 `--upgrade` 来确保您使用的是 EB CLI 的最新版本：

```
$ sudo pip install --upgrade awsebcli
```

有关完整的 EB CLI 安装说明，请参阅安装 EB CLI。

要使用 EB CLI 监控您的环境的运行状况，您必须首先通过运行 eb init 并按照提示操作来配置本地项目文件夹。有关完整说明，请参阅配置 EB CLI。

如果您已有一个在 Elastic Beanstalk 中运行的环境并且想使用 EB CLI 监控其运行状况，请执行这些步骤以将其附加到现有环境。

将 EB CLI 附加到现有环境

1. 打开命令行终端并导航至您的用户文件夹。
2. 为您的环境创建一个新的文件夹并打开该文件夹。
3. 运行 eb init 命令，然后选择要监控其运行状况的应用程序和环境。如果您只有一个运行您选择的应用程序的环境，EB CLI 将自动选择它，您无需选择环境，如以下示例所示。

```
~/project$ eb init
Select an application to use
1) elastic-beanstalk-example
2) [ Create new Application ]
(default is 2): 1
Select the default environment.
You can change this later by typing "eb use [environment_name]".
1) elasticBeanstalkEx2-env
2) elasticBeanstalkExa-env
(default is 1): 1
```

使用 EB CLI 监控运行状况

1. 打开命令行并导航到项目文件夹。
2. 运行 eb health 命令以显示您的环境中的实例的运行状况。在此示例中，有五个实例在 Linux 环境中运行。

```
~/project $ eb health
 elasticBeanstalkExa-env                                Ok
 2015-07-08 23:13:20
WebServer
    Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info    pending  unknown
    5        5        0        0        0        0        0        0

  instance-id   status     cause
                                          health
```

```
   Overall     Ok
 i-d581497d    Ok
 i-d481497c    Ok
 i-136e00c0    Ok
 i-126e00c1    Ok
 i-8b2cf575    Ok

 instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75
p50     p10                                 requests
   Overall     671.8   100.0    0.0    0.0    0.0    0.003    0.002    0.001
0.001   0.000
 i-d581497d    143.0    1430      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-d481497c    128.8    1288      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-136e00c0    125.4    1254      0      0      0    0.004    0.002    0.001
0.001   0.000
 i-126e00c1    133.4    1334      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-8b2cf575    141.2    1412      0      0      0    0.003    0.002    0.001
0.001   0.000

 instance-id   type       az   running     load 1  load 5      user%  nice%
system%  idle%   iowait%                             cpu
 i-d581497d    t2.micro   1a   12 mins        0.0    0.04        6.2    0.0
1.0   92.5       0.1
 i-d481497c    t2.micro   1a   12 mins       0.01    0.09        5.9    0.0
1.6   92.4       0.1
 i-136e00c0    t2.micro   1b   12 mins       0.15    0.07        5.5    0.0
0.9   93.2       0.0
 i-126e00c1    t2.micro   1b   12 mins       0.17    0.14        5.7    0.0
1.4   92.7       0.1
 i-8b2cf575    t2.micro   1c   1 hour        0.19    0.08        6.5    0.0
1.2   92.1       0.1

 instance-id   status     id   version             ago
                                      deployments
 i-d581497d    Deployed   1    Sample Application   12 mins
 i-d481497c    Deployed   1    Sample Application   12 mins
 i-136e00c0    Deployed   1    Sample Application   12 mins
 i-126e00c1    Deployed   1    Sample Application   12 mins
 i-8b2cf575    Deployed   1    Sample Application   1 hour
```

在此示例中，有一个实例在 Windows 环境中运行。

```
~/project $ eb health
 WindowsSampleApp-env                                Ok
      2018-05-22 17:33:19
WebServer                                             IIS 10.0 running on 64bit
 Windows Server 2016/2.2.0
  total      ok    warning  degraded  severe    info   pending  unknown
    1        1        0        0        0        0        0        0

  instance-id           status     cause
                                          health
    Overall             Ok
  i-065716fba0e08a351   Ok

  instance-id           r/sec    %2xx   %3xx   %4xx   %5xx       p99       p90
 p75     p50     p10                          requests
    Overall              13.7   100.0    0.0    0.0    0.0    1.403    0.970
 0.710   0.413   0.079
  i-065716fba0e08a351     2.4   100.0    0.0    0.0    0.0    1.102*   0.865
 0.601   0.413   0.091

  instance-id           type       az   running     % user time    % privileged
 time  % idle time                                  cpu
  i-065716fba0e08a351   t2.large   1b   4 hours             0.2
 0.1         99.7

  instance-id           status     id   version              ago
                                      deployments
  i-065716fba0e08a351   Deployed   2    Sample Application   4 hours
```

读取输出

输出将在屏幕顶部显示环境的名称、环境的总体运行状况和当前日期。

```
elasticBeanstalkExa-env                                Ok
 2015-07-08 23:13:20
```

接下来三行将显示环境类型（本例中为“WebServer”）、配置 (Ruby 2.1 with Puma) 以及每种状态（共七种状态）的实例数量的明细。

```
WebServer
 Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info    pending  unknown
    5        5        0        0        0        0        0        0
```

输出的其余部分分为四个部分。第一个部分依次显示了环境总体和每个实例的状态 和状态原因。以下示例显示了环境中两个状态为 Info 的实例以及一个表示部署已开始的原因。

```
  instance-id    status     cause
                                    health
    Overall      Ok
  i-d581497d     Info       Performing application deployment (running for 3 seconds)
  i-d481497c     Info       Performing application deployment (running for 3 seconds)
  i-136e00c0     Ok
  i-126e00c1     Ok
  i-8b2cf575     Ok
```

有关运行状况的状态和颜色的信息，请参阅运行状况颜色和状态。

requests 部分显示了 Web 服务器日志中有关每个实例的信息。在此示例中，每个实例正常接收请求且没有错误。

```
  instance-id    r/sec     %2xx    %3xx    %4xx    %5xx       p99       p90       p75      p50
    p10                                 requests
    Overall      13.7    100.0     0.0     0.0     0.0     1.403     0.970     0.710    0.413
  0.079
  i-d581497d     2.4     100.0     0.0     0.0     0.0     1.102*    0.865     0.601    0.413
  0.091
  i-d481497c     2.7     100.0     0.0     0.0     0.0     0.842*    0.788     0.480    0.305
  0.062
  i-136e00c0     4.1     100.0     0.0     0.0     0.0     1.520*    1.088     0.883    0.524
  0.104
  i-126e00c1     2.2     100.0     0.0     0.0     0.0     1.334*    0.791     0.760    0.344
  0.197
  i-8b2cf575     2.3     100.0     0.0     0.0     0.0     1.162*    0.867     0.698    0.477
  0.076
```

cpu 部分显示了每个实例的操作系统指标。输出因操作系统而异。以下是 Linux 环境的输出。

```
  instance-id   type       az   running     load 1  load 5      user%  nice%  system%
 idle%   iowait%                            cpu
```

```
 i-d581497d    t2.micro   1a   12 mins        0.0    0.03        0.2    0.0      0.0
99.7       0.1
 i-d481497c    t2.micro   1a   12 mins        0.0    0.03        0.3    0.0      0.0
99.7       0.0
 i-136e00c0    t2.micro   1b   12 mins        0.0    0.04        0.1    0.0      0.0
99.9       0.0
 i-126e00c1    t2.micro   1b   12 mins       0.01    0.04        0.2    0.0      0.0
99.7       0.1
 i-8b2cf575    t2.micro   1c   1 hour         0.0    0.01        0.2    0.0      0.1
99.6       0.1
```

以下是 Windows 环境的输出。

```
 instance-id           type       az   running     % user time    % privileged time  %
idle time
 i-065716fba0e08a351   t2.large   1b   4 hours             0.2                  0.0
    99.8
```

有关显示的服务器和操作系统指标的信息，请参阅实例指标。

最后的 deployments 部分显示了每个实例的部署状态。如果滚动部署失败，您可以使用显示的部署 ID、状态和版本标签来确定环境中正在运行错误版本的实例。

```
  instance-id   status     id   version              ago
                                 deployments
  i-d581497d    Deployed   1    Sample Application   12 mins
  i-d481497c    Deployed   1    Sample Application   12 mins
  i-136e00c0    Deployed   1    Sample Application   12 mins
  i-126e00c1    Deployed   1    Sample Application   12 mins
  i-8b2cf575    Deployed   1    Sample Application   1 hour
```

交互式运行状况视图

eb health 命令将显示环境运行状况的快照。要每 10 秒将显示的信息刷新一次，请使用 `--refresh` 选项。

```
$ eb health --refresh
 elasticBeanstalkExa-env                          Ok
 2015-07-09 22:10:04 (1 secs)
WebServer
          Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
```

```
    5        5        0        0        0        0        0        0

 instance-id   status     cause
                                    health
   Overall     Ok
 i-bb65c145    Ok         Application deployment completed 35 seconds ago and took 26
seconds
 i-ba65c144    Ok         Application deployment completed 17 seconds ago and took 25
seconds
 i-f6a2d525    Ok         Application deployment completed 53 seconds ago and took 26
seconds
 i-e8a2d53b    Ok         Application deployment completed 32 seconds ago and took 31
seconds
 i-e81cca40    Ok

 instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75     p50
  p10                              requests
   Overall     671.8   100.0    0.0    0.0    0.0    0.003    0.002    0.001   0.001
0.000
 i-bb65c145    143.0    1430      0      0      0    0.003    0.002    0.001   0.001
0.000
 i-ba65c144    128.8    1288      0      0      0    0.003    0.002    0.001   0.001
0.000
 i-f6a2d525    125.4    1254      0      0      0    0.004    0.002    0.001   0.001
0.000
 i-e8a2d53b    133.4    1334      0      0      0    0.003    0.002    0.001   0.001
0.000
 i-e81cca40    141.2    1412      0      0      0    0.003    0.002    0.001   0.001
0.000

 instance-id   type       az   running     load 1  load 5      user%  nice%  system%
idle%   iowait%                            cpu
 i-bb65c145    t2.micro   1a   12 mins        0.0    0.03        0.2    0.0      0.0
99.7       0.1
 i-ba65c144    t2.micro   1a   12 mins        0.0    0.03        0.3    0.0      0.0
99.7       0.0
 i-f6a2d525    t2.micro   1b   12 mins        0.0    0.04        0.1    0.0      0.0
99.9       0.0
 i-e8a2d53b    t2.micro   1b   12 mins       0.01    0.04        0.2    0.0      0.0
99.7       0.1
 i-e81cca40    t2.micro   1c   1 hour         0.0    0.01        0.2    0.0      0.1
99.6       0.1
```

```
  instance-id   status     id   version              ago
                                  deployments
  i-bb65c145    Deployed   1    Sample Application   12 mins
  i-ba65c144    Deployed   1    Sample Application   12 mins
  i-f6a2d525    Deployed   1    Sample Application   12 mins
  i-e8a2d53b    Deployed   1    Sample Application   12 mins
  i-e81cca40    Deployed   1    Sample Application   1 hour

 (Commands: Help,Quit, # # # #)
```

此示例显示了一个最近从一个实例扩展到五个实例的环境。扩展操作成功完成，现在，所有实例正在通过运行状况检查并已准备好接收请求。在交互式模式中，运行状况的状态每 10 秒更新一次。在右上角，计时器为下一次更新进行倒计时。

在左下角，报告显示了一个选项列表。要退出交互模式，请按 Q。要滚动，请按箭头键。要查看其他命令的列表，请按 H。

交互运行状况视图选项

在以交互方式查看环境运行状况时，您可以使用键盘键调整视图并告知 Elastic Beanstalk 替换或重新启动各个实例。要在以交互模式查看运行状况报告时查看可用命令的列表，请按 H。

```
  up,down,home,end   Scroll vertically
  left,right         Scroll horizontally
  F                  Freeze/unfreeze data
  X                  Replace instance
  B                  Reboot instance
  <,>                Move sort column left/right
  -,+                Sort order descending/ascending
  P                  Save health snapshot data file
  Z                  Toggle color/mono mode
  Q                  Quit this program

  Views
  1                  All tables/split view
  2                  Status Table
  3                  Request Summary Table
  4                  CPU%/Load Table
  H                  This help menu


(press Q or ESC to return)
```

使用 EB CLI 以组的形式管理多个 Elastic Beanstalk 环境

您可以使用 EB CLI 创建一组 AWS Elastic Beanstalk 环境，每个环境运行面向服务的架构应用程序的一个单独组件。EB CLI 使用 ComposeEnvironments API 管理此类组。

> **Note**
>
> 环境组不同于多容器 Docker 环境中的多个容器。使用环境组，您应用程序的各个组件运行在单独的 Elastic Beanstalk 环境中，具有自己的专用 Amazon EC2 实例集。每个组件可以单独扩展。借助多容器 Docker，您可以将应用程序的多个组件组合成单个环境。所有组件共享相同的一组 Amazon EC2 实例，每个实例运行多个 Docker 容器。根据应用程序的需求，选择这些架构之一。
>
> 有关多容器 Docker 的详细信息，请参阅使用 Amazon ECS 平台分支。

将您的应用程序组件组织到以下文件夹结构中：

```
~/project-name
|-- component-a
|   `-- env.yaml
`-- component-b
    `-- env.yaml
```

在每个子文件夹中，都包含在自有环境中运行的独立应用程序组件的源代码，以及一个名为 `env.yaml` 的环境定义文件。有关 `env.yaml` 格式的详细信息，请参阅环境清单 (env.yaml)。

要使用 `Compose Environments` API，首先要从项目文件夹中运行 eb init，同时利用 `--modules` 选项通过各组件所在文件夹的名称来指定每个组件。

```
~/workspace/project-name$ eb init --modules component-a component-b
```

EB CLI 会提示您配置每个组件，然后在各组件文件夹中创建 `.elasticbeanstalk` 目录。EB CLI 不会在父目录中创建配置文件。

```
~/project-name
|-- component-a
|   |-- .elasticbeanstalk
|   `-- env.yaml
`-- component-b
```

```
    |-- .elasticbeanstalk
    `-- env.yaml
```

然后，使用要创建的环境的列表运行 eb create 命令，每个组件对应一个环境：

```
~/workspace/project-name$ eb create --modules component-a component-b --env-group-suffix group-name
```

此命令会为每个组件创建一个环境。环境名是用连字符将 `EnvironmentName` 文件中指定的 `env.yaml` 与组名串联起来得到的。这两个选项加连字符的总长度不得超过 23 个字符的环境名最大长度限制。

要更新环境，请使用 eb deploy 命令：

```
~/workspace/project-name$ eb deploy --modules component-a component-b
```

您可以单独更新每个组件，也可以成组更新。可使用 `--modules` 选项指定要更新的组件。

EB CLI 将您随 eb create 使用的组名称存储在 `branch-defaults` 下的 EB CLI 配置文件的 `/.elasticbeanstalk/config.yml` 部分中。要将您的应用程序部署到不同的组，请在运行 `--env-group-suffix` 时使用 eb deploy 选项。如果指定的组不存在，EB CLI 会创建一个新的环境组。

```
~/workspace/project-name$ eb deploy --modules component-a component-b --env-group-suffix group-2-name
```

要终止环境，请在每个模块的文件夹中运行 eb terminate。默认情况下，如果您尝试终止另一正在运行的环境所依赖的环境，EB CLI 会显示一条错误。请先终止依赖环境，或使用 `--ignore-links` 选项覆盖默认行为。

```
~/workspace/project-name/component-b$ eb terminate --ignore-links
```

解决 EB CLI 方面的问题

本主题列出了使用 EB CLI 时遇到的常见错误消息和可能的解决方案。如果您遇到此处未显示的错误消息，请使用反馈 链接告诉我们。

ERROR: An error occurred while handling git command.Error code: 128 Error: fatal: Not a valid object name HEAD

原因：初始化 Git 存储库后但尚未提交时会显示此错误消息。当项目文件夹包含 Git 存储库时，EB CLI 会查找 HEAD 修订。

解决方案：将项目文件夹中的文件添加到暂存区中并提交。

```
~/my-app$ git add .
~/my-app$ git commit -m "First commit"
```

ERROR: This branch does not have a default environment.You must either specify an environment by typing "eb status my-env-name" or set a default environment by typing "eb use my-env-name".

原因：在 git 中创建新分支时，默认情况下未挂载到 Elastic Beanstalk 环境。

解决方案：运行 eb list 查看可用环境列表。然后运行 eb use ***env-name*** 使用一个可用环境。

ERROR: 2.0+ Platforms require a service role.You can provide one with --service-role option

原因：如果您使用 eb create 指定环境名称（例如 eb create my-env），EB CLI 不会尝试为您创建服务角色。如果没有默认服务角色，就会显示以上错误。

解决方案：不带环境名称运行 eb create，按照提示创建默认服务角色。

部署故障排除

如果您的 Elastic Beanstalk 部署没有按计划顺利运行，您可能收到 404（应用程序无法启动时）或 500（应用程序在运行时失败）响应而不是看到您的网站。要排除众多常见问题，您可以使用 EB CLI 检查部署的状态、查看其日志、使用 SSH 获取对 EC2 实例的访问，或者打开您的应用程序环境的 AWS管理控制台页面。

使用 EB CLI 帮助排除部署的问题

1. 运行 eb status 以查看当前部署的状态和 EC2 主机的运行状况。例如：

```
$ eb status --verbose

Environment details for: python_eb_app
  Application name: python_eb_app
  Region: us-west-2
  Deployed Version: app-150206_035343
  Environment ID: e-wa8u6rrmqy
  Platform: 64bit Amazon Linux 2014.09 v1.1.0 running Python 2.7
```

```
  Tier: WebServer-Standard-
  CNAME: python_eb_app.elasticbeanstalk.com
  Updated: 2015-02-06 12:00:08.557000+00:00
  Status: Ready
  Health: Green
  Running instances: 1
      i-8000528c: InService
```

> **Note**
>
> 使用 --verbose 开关可提供有关您正在运行实例状态的信息。如果不使用它，eb status 将只输出有关您的环境的一般信息。

2. 运行 eb health 以查看有关您的环境的运行状况信息：

```
$ eb health --refresh
 elasticBeanstalkExa-env                                    Degraded
 2016-03-28 23:13:20
WebServer
    Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5        2        0        2        1        0        0        0

  instance-id   status     cause
    Overall     Degraded  Incorrect application version found on 3 out of 5
 instances. Expected version "Sample Application" (deployment 1).
  i-d581497d    Degraded  Incorrect application version "v2" (deployment 2).
 Expected version "Sample Application" (deployment 1).
  i-d481497c    Degraded  Incorrect application version "v2" (deployment 2).
 Expected version "Sample Application" (deployment 1).
  i-136e00c0    Severe    Instance ELB health has not been available for 5 minutes.
  i-126e00c1    Ok
  i-8b2cf575    Ok

  instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75
 p50     p10
    Overall     646.7   100.0    0.0    0.0    0.0    0.003    0.002    0.001
 0.001   0.000
  i-dac3f859    167.5    1675      0      0      0    0.003    0.002    0.001
 0.001   0.000
  i-05013a81    161.2    1612      0      0      0    0.003    0.002    0.001
 0.001   0.000
```

```
 i-04013a80    0.0          -      -      -      -          -        -       -
-       -
 i-3ab524a1    155.9    1559      0      0      0    0.003    0.002    0.001
0.001   0.000
 i-bf300d3c    162.1    1621      0      0      0    0.003    0.002    0.001
0.001   0.000

 instance-id   type        az   running     load 1  load 5      user%  nice%
system%  idle%   iowait%
 i-d581497d    t2.micro    1a   25 mins       0.16     0.1        7.0    0.0
1.7   91.0       0.1
 i-d481497c    t2.micro    1a   25 mins       0.14     0.1        7.2    0.0
1.6   91.1       0.0
 i-136e00c0    t2.micro    1b   25 mins        0.0    0.01        0.0    0.0
0.0   99.9       0.1
 i-126e00c1    t2.micro    1b   25 mins       0.03    0.08        6.9    0.0
2.1   90.7       0.1
 i-8b2cf575    t2.micro    1c   1 hour        0.05    0.41        6.9    0.0
2.0   90.9       0.0

 instance-id   status     id   version              ago
     deployments
 i-d581497d    Deployed   2    v2                   9 mins
 i-d481497c    Deployed   2    v2                   7 mins
 i-136e00c0    Failed     2    v2                   5 mins
 i-126e00c1    Deployed   1    Sample Application   25 mins
 i-8b2cf575    Deployed   1    Sample Application   1 hour
```

以上示例显示带有五个实例的环境，第三个实例上的版本“v2”部署失败。在部署失败之后，预期版本将重置为最后一个成功的版本，在本例中是第一个部署的“Sample Application”。有关更多信息，请参阅使用 EB CLI 监控环境的运行状况。

3. 运行 eb logs 以下载和查看与您的应用程序部署关联的日志。

```
$ eb logs
```

4. 运行 eb ssh 以连接在应用程序上运行的 EC2 实例并直接查看它。在实例上，您部署的应用程序可在 /opt/python/current/app 目录中找到，您的 Python 环境可在 /opt/python/run/venv/ 中找到。

5. 运行 eb console 以在AWS管理控制台上查看您的应用程序环境。您可以使用 Web 界面方便地检查部署的各个方面，包括应用程序的配置、状态、事件和日志。您也可以下载当前或过去您已部署到服务器的应用程序版本。

EB CLI 命令参考

您可以使用 Elastic Beanstalk 命令行界面（EB CLI）执行多种操作来部署和管理您的 Elastic Beanstalk 应用程序和环境。如果您要部署 Git 源代码控制下的应用程序源代码，可将 EB CLI 与 Git 集成。有关更多信息，请参阅使用 Elastic Beanstalk 命令行界面 (EB CLI)和将 EB CLI 与 Git 配合使用。

命令

- eb abort
- eb appversion
- eb clone
- eb codesource
- eb config
- eb console
- eb create
- eb deploy
- eb events
- eb health
- eb init
- eb labs
- eb list
- eb local
- eb logs
- eb open
- eb platform
- eb printenv
- eb restore
- eb scale
- eb setenv
- eb ssh
- eb status
- eb swap

- eb tags
- eb terminate
- eb upgrade
- eb use
- 常用选项

eb abort

描述

当针对实例的环境配置更改仍在进行中时取消升级。

> **Note**
> 如果您有两个以上的正在进行更新的环境，系统将提示您选择要为其回滚更改的环境的名称。

语法

eb abort

eb abort ***environment-name***

选项

名称	描述
常用选项	

输出

该命令显示当前正在更新的环境的列表，并提示您选择要中止的更新。如果当前仅在更新一个环境，则无需指定环境名称。如果成功，该命令将恢复环境配置更改。回滚过程将继续，直至环境中的所有实例拥有上一个环境的配置或回滚过程失败。

示例

以下示例取消平台升级。

```
$ eb abort
Aborting update to environment "tmp-dev".
<list of events>
```

eb appversion

描述

EB CLI appversion 命令管理 Elastic Beanstalk 应用程序版本。您可以在无部署的情况下创建应用程序的新版本、删除应用程序的版本或创建应用程序版本生命周期。如果您不带任何选项调用命令，它将进入交互模式。

使用 --create 选项可创建应用程序的新版本。

使用 --delete 选项可删除应用程序的某个版本。

使用 lifecycle 选项可显示或创建应用程序的版本生命周期策略。有关更多信息，请参阅 the section called “版本生命周期”。

语法

eb appversion

eb appversion [-c | --create]

eb appversion [-d | --delete] ***version-label***

eb appversion lifecycle [-p | --print]

选项

名称	描述 类型：字符串
-a *application-name* 或 --application_name *application-name*	应用程序的名称。如果未找到具有指定名称的应用程序，则 EB CLI 会为新应用程序创建应用程序版本。 仅适用于 --create 选项。 类型：字符串

名称	描述 类型：字符串
-c 或 --create	创建应用程序的新版本。
-d *version-label* 或 --delete *version-label*	删除标签为 *version-label* 的应用程序版本。
-l *version_label* 或 --label *version_label*	指定要用作 EB CLI 所创建版本的标签。如果您不使用此选项，EB CLI 将生成新的唯一标签。如果您提供了版本标签，请确保它是唯一的。 仅适用于 --create 选项。 类型：字符串
生命周期	调用默认编辑器以创建新的应用程序版本生命周期策略。使用此策略可避免达到应用程序版本配额。
lifecycle -p 或 lifecycle --print	显示当前应用程序生命周期策略。
-m "*version_description* " 或 --message "*version_description* "	应用程序版本的描述。它用双引号引起来。 仅适用于 --create 选项。 类型：字符串

名称 类型：字符串	描述
-p 或 --process	预处理并验证源代码包中的环境清单和配置文件。验证配置文件可以识别问题。我们建议您在将应用程序版本部署到此环境中之前执行此操作。 仅适用于 --create 选项。
--source codecommit/ *repository-name*/*branch-name*	CodeCommit 存储库和分支。有关更多信息，请参阅通过 AWS CodeCommit 使用 EB CLI。 仅适用于 --create 选项。
--staged	使用 git 索引中暂存的文件（而不是 HEAD 提交）来创建应用程序版本。 仅适用于 --create 选项。
--timeout *minutes*	命令超时之前的分钟数。 仅适用于 --create 选项。
常用选项	

以交互方式使用命令

如果您使用不带任何参数的命令，则将显示应用程序的版本。它们按时间倒序方式列出，最新版本列在首位。有关屏幕示例，请参阅 Examples（示例）部分。注意显示在底部的状态行。状态行显示上下文相关信息。

按 d 可删除应用程序版本，按 l 可管理应用程序的生命周期策略，按 q 可退出且不保存任何更改。

> **Note**
>
> 如果版本部署到了任何环境，则您无法删除此版本。

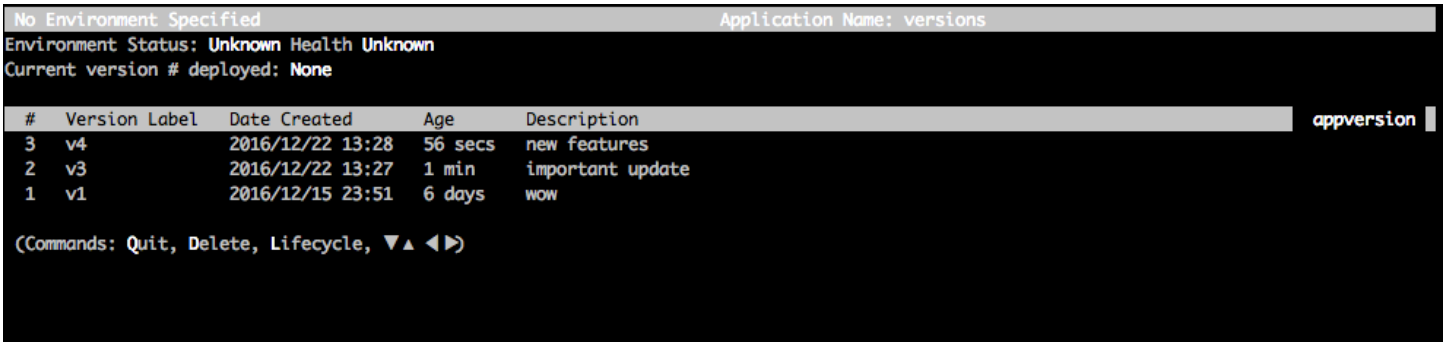
输出

带有 `--create` 选项的命令会显示一条消息，确认应用程序版本已创建。

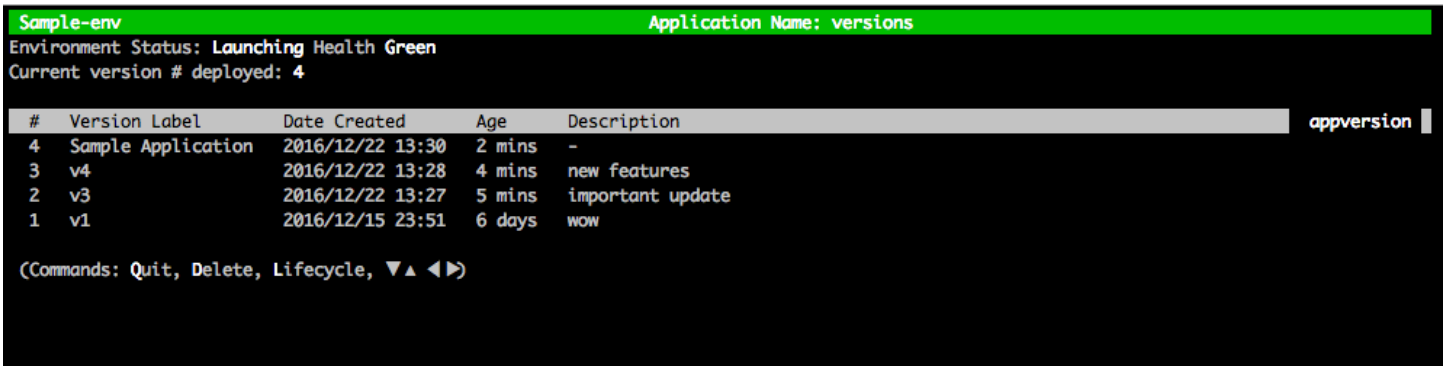
带有 `--delete` *`version-label`* 选项的命令会显示一条消息，确认应用程序版本已删除。

示例

下面的示例显示了未进行任何部署的应用程序的交互式窗口。



下面的示例显示了已部署第四个版本、版本标签为 Sample Application（示例应用程序）的应用程序的交互式窗口。



以下示例显示 eb appversion lifecycle -p 命令的输出，其中 *`ACCOUNT-ID`* 是用户的账户 ID：

```
Application details for: lifecycle
  Region: sa-east-1
  Description: Application created from the EB CLI using "eb init"
  Date Created: 2016/12/20 02:48 UTC
  Date Updated: 2016/12/20 02:48 UTC
  Application Versions: ['Sample Application']
  Resource Lifecycle Config(s):
    VersionLifecycleConfig:
      MaxCountRule:
        DeleteSourceFromS3: False
```

```
    Enabled: False
    MaxCount: 200
  MaxAgeRule:
    DeleteSourceFromS3: False
    Enabled: False
    MaxAgeInDays: 180
ServiceRole: arn:aws:iam::ACCOUNT-ID:role/aws-elasticbeanstalk-service-role
```

eb clone

描述

将一个环境克隆到一个新环境，这样两个环境将具有相同的环境设置。

> **Note**
>
> 默认情况下，无论您从中创建克隆的环境的解决方案堆栈版本如何，eb clone 命令都会使用最新的解决方案堆栈创建克隆环境。您可以在运行此命令时包括 `--exact` 选项来禁止此操作。

> **Important**
>
> 克隆的 Elastic Beanstalk 环境不会延续安全组进行入口，从而使环境对所有互联网流量开放。您需要为克隆的环境重新建立入口安全组。
> 您可以通过检查环境配置的偏差状态来查看可能无法克隆的资源。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的检测整个 CloudFormation 堆栈上的偏差。

语法

eb clone

eb clone ***environment-name***

Options

名称	描述
-n *string*	克隆环境的所需名称。

名称	描述
或 `--clone_name` *string*	
`-c` *string* 或 `--cname` *string*	克隆环境的所需别名记录前缀。
`--envvars`	逗号分隔列表中的环境属性，格式为 *name*=*value*。 类型：字符串 约束： • 必须用逗号分隔键值对。 • 键和值可以包含任何语言的任何字母字符、任何数字字符、空格、不可见的分隔符和以下符号：_ . : / + \ - @ • 键最多可包含 128 个字符。值最多可包含 256 个字符。 • 键和值区分大小写。 • 值不能与环境名称匹配。 • 值不能包含 `aws:` 或 `elasticbeanstalk:` 。 • 所有环境属性的组合大小不得超过 4096 字节。
`--exact`	防止 Elastic Beanstalk 将新克隆环境的解决方案堆栈版本更新为可用的最新版本（适用于原始环境的平台）。
`--scale` *number*	在克隆环境启动时要在其中运行的实例的数目。
`--tags` *name*=*value*	逗号分隔列表中的环境的资源标签，格式为 *name*=*value*。
`--timeout`	命令超时之前的分钟数。
常用选项	

输出

如果成功，此命令将创建一个具有与原始环境相同的设置或具有由任何 eb clone 选项指定的环境修改的环境。

示例

以下示例克隆指定环境。

```
$ eb clone
Enter name for Environment Clone
(default is tmp-dev-clone):
Enter DNS CNAME prefix
(default is tmp-dev-clone):
Environment details for: tmp-dev-clone
  Application name: tmp
  Region: us-west-2
  Deployed Version: app-141029_144740
  Environment ID: e-vjvrqnn5pv
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev-clone.elasticbeanstalk.com
  Updated: 2014-10-29 22:00:23.008000+00:00
Printing Status:
2018-07-11 21:04:20    INFO: createEnvironment is starting.
2018-07-11 21:04:21    INFO: Using elasticbeanstalk-us-west-2-888888888888 as Amazon S3
 storage bucket for environment data.
...
2018-07-11 21:07:10    INFO: Successfully launched environment: tmp-dev-clone
```

eb codesource

描述

将 EB CLI 配置为从 CodeCommit 存储库进行部署，或禁用 CodeCommit 集成并从本地计算机上传源包。

> **Note**
>
> 有些 AWS 区域不提供 CodeCommit。Elastic Beanstalk 和 CodeCommit 之间的集成在这些区域无法正常工作。

有关每个区域中提供的 AWS 服务的信息，请参阅区域表。

语法

eb codesource

eb codesource codecommit

eb codesource local

选项

名称	描述
常用选项	

输出

eb codesource 会提示您在 CodeCommit 集成和标准部署之间选择。

eb codesource codecommit 启动 CodeCommit 集成的交互式存储库配置。

eb codesource local 显示初始配置并禁用 CodeCommit 集成。

示例

使用 eb codesource codecommit 可配置当前分支的 CodeCommit 集成。

```
~/my-app$ eb codesource codecommit
Select a repository
1) my-repo
2) my-app
3) [ Create new Repository ]
(default is 1): 1

Select a branch
1) mainline
2) test
3) [ Create new Branch with local HEAD ]
(default is 1): 1
```

使用 eb codesource local 可禁用当前分支的 CodeCommit 集成。

```
~/my-app$ eb codesource local
Current CodeCommit setup:
  Repository: my-app
  Branch: mainline
Default set to use local sources
```

eb config

描述

管理活动配置设置和您的环境保存的配置。您可以使用此命令上载、下载或列出环境中已保存的配置。您还可以使用它来下载、显示或更新其活动配置设置。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令还会更改生成器配置设置。这是基于 `platform.yaml` 中设置的值来完成的。

> **Note**
>
> eb config 不会显示环境属性。要设置可在您的应用程序中读取的环境属性，请使用 eb setenv。

语法

以下是 eb config 命令用于使用环境的活动配置设置的部分语法。有关具体示例，请参阅查看本主题后面的示例部分。

- eb config – 在文本编辑器中显示您配置为 EDITOR 环境变量的环境的活动配置设置。当您保存对文件所做的更改并关闭编辑器时，将使用您保存在文件中的选项设置来更新环境。

  > **Note**
  >
  > （如果您尚未配置 EDITOR 环境变量，则 EB CLI 将在默认编辑器中显示 YAML 文件的选项设置。）

- eb config ***environment-name*** – 显示并更新命名环境的配置。配置显示在您配置的文本编辑器中或默认编辑器 YAML 文件中。

- eb config save – 使用文件名 `[configuration-name].cfg.yml` 将当前环境的活动配置设置保存到 `.elasticbeanstalk/saved_configs/`。默认情况下，EB CLI 将基于环境名称使用 *`configuration-name`* 保存配置设置。可以在运行此命令时将 `--cfg` 选项与所需的配置名称一起包含来指定其他配置名称。

  您可以使用 `--tags` 选项标记保存的配置。
- eb config **`--display`** – 将环境的活动配置设置写入 stdout（而非文件）。默认情况下，这将显示终端的配置设置。
- eb config **`--update`** ***`configuration_string`*** **`|`** ***`file_path`*** – 使用 *`configuration_string`* 中指定的信息或由 *`file_path`* 标识的文件中指定的信息更新当前环境的活动配置设置。

> **Note**
>
> `--display` 和 `--update` 选项提供了以编程方式读取和修改环境配置设置的灵活性。

下面介绍了使用 eb config 命令处理已保存的配置时遵循的语法。有关示例，请参阅查看本主题后面的示例部分。

- eb config get ***`config-name`*** – 从 Amazon S3 下载命名的已保存配置。
- eb config delete ***`config-name`*** – 从 Amazon S3 删除命名的已保存配置。如果您已经下载，也会在本地删除它。
- eb config list - 列出您在 Amazon S3 中拥有的已保存配置。
- eb config put ***`filename`*** - 将命名的已保存配置上传到 Amazon S3 存储桶。*`filename`* 的文件扩展名必须为 `.cfg.yml`。要指定不带路径的文件名，可以先将文件保存到 `.elasticbeanstalk` 文件夹或 `.elasticbeanstalk/saved_configs/` 文件夹，然后再运行此命令。或者，您可以通过提供完整路径来指定 *`filename`*。

选项

名称	描述
`--cfg` *`config-name`*	用于已保存配置的名称。

名称	描述
	此选项仅适用于 eb config save。

`-d`
 	
或
 	
`--display`
	显示当前环境的配置设置（写入 stdout)。
	
	将与 `--format` 选项结合使用以指定 JSON 或 YAML 格式的输出。如果不指定输出格式，将使用 YAML 格式。
	
	此选项仅在您使用 eb config 命令（而不使用任何其他子命令）时适用。

`-f` *`format_type`* 或 `--format` *`format_type`*	指定显示格式。有效值包括 JSON 或 YAML。 默认值为 YAML。 此选项仅适用于 `--display` 选项。
`--tags` *`key1`*`=`*`value1`*`[,`*`ke`	要添加到保存的配置的标签。在列表中指定标签时，将其指定为键 = 值对，并用逗号分隔每个标签。 有关更多信息，请参阅标记保存的配置。 此选项仅适用于 eb config save。
`--timeout` *`timeout`*	命令超时之前的分钟数。

名称	描述

`-u` *configuration_string* | *file_path*
 	
或
 	
`--update` *configuration_string* | *file_path*
	更新当前环境的活动配置设置。
	
	此选项仅在您使用 eb config 命令（而不使用任何其他子命令）时适用。
	
	configuration_string | *file_path* 参数的类型为字符串。字符串提供了要从环境配置设置中添加、更新或删除的命名空间列表和相应选项。或者，输入字符串可以表示包含相同信息的文件。
	
	要指定文件名，输入字符串必须遵循 "file://<*path*><*filename*>" 格式。要指定不带 *path* 的文件名，可以将文件保存运行命名的文件夹中。或者，您可以通过提供完整路径来指定 filename。
	
	配置信息必须符合以下条件。至少需要 OptionSettings 或 OptionsToRemove 部分之一。使用 OptionSettings 添加或更改选项。使用 OptionsToRemove 从命名空间中删除选项。有关具体示例，请参阅查看本主题后面的示例部分。
	
	Example
	
	YAML 格式
	
	`OptionSettings:`
	`  namespace1:`
	`    option-name-1:` *option-value-1*
	`    option-name-2:` *option-value-2*
	`    ...`
	`OptionsToRemove:`
	`  namespace1:`
	`    option-name-1`
	`    option-name-2`
	`    ...`
	
	Example
	
	JSON 格式

名称	描述

	<pre>{
	  "OptionSettings": {
	  "namespace1": {
	  "option-name-1": " *option-value-1* ",
	  "option-name-2": " *option-value-2* ",
	  ...
	  }
	  },
	  "OptionsToRemove": {
	  "namespace1": {
	  "option-name-1",
	  "option-name-2",
	  ...
	  }
	  }
	}</pre>

常用选项	

输出

如果 eb config 或 eb config ***environment-name*** 命令在无子命令或不添加选项的情况下运行成功，则将在文本编辑器中显示您配置为 EDITOR 环境变量的当前的选项设置。（如果您尚未配置 EDITOR 环境变量，则 EB CLI 将在默认编辑器中显示 YAML 文件的选项设置。）

当您保存对文件所做的更改并关闭编辑器时，将使用您保存在文件中的选项设置来更新环境。将显示以下输出以确认配置更新。

```
$ eb config myApp-dev
    Printing Status:
    2021-05-19 18:09:45    INFO    Environment update is starting.
    2021-05-19 18:09:55    INFO    Updating environment myApp-dev's configuration
 settings.
    2021-05-19 18:11:20    INFO    Successfully deployed new configuration to
 environment.
```

如果使用 `--display` 选项成功运行命令，它会显示当前环境的配置设置（写入 stdout)。

如果此命令在具有 `get` 参数的情况下成功运行，则将显示您下载的本地副本的位置。

如果此命令在具有 save 参数的情况下成功运行，则将显示已保存文件的位置。

示例

本部分介绍如何更改用来查看和编辑选项设置文件的文本编辑器。

对于 Linux 和 UNIX，以下示例将编辑器更改为 vim：

```
$ export EDITOR=vim
```

对于 Linux 和 UNIX，以下示例将编辑器更改为安装在 /usr/bin/kate 中的项。

```
$ export EDITOR=/usr/bin/kate
```

对于 Windows，以下示例将编辑器更改为 Notepad++。

```
> set EDITOR="C:\Program Files\Notepad++\Notepad++.exe
```

本部分提供了针对 eb config 命令（当此命令在具有子命令的情况下运行时）的示例。

以下示例删除名为 app-tmp 的已保存配置。

```
$ eb config delete app-tmp
```

以下示例从 Amazon S3 存储桶下载名为 app-tmp 的已保存配置。

```
$ eb config get app-tmp
```

以下示例列出存储在 Amazon S3 存储桶中的已保存配置的名称。

```
$ eb config list
```

以下示例将名为 app-tmp 的已保存配置的本地副本上传到 Amazon S3 存储桶。

```
$ eb config put app-tmp
```

以下示例保存来自当前运行的环境的配置设置。如果您没有提供用于已保存配置的名称，Elastic Beanstalk 将根据环境名称命名配置文件。例如，名为 tmp-dev 的环境将被称为 tmp-

dev.cfg.yml。Elastic Beanstalk 将文件保存到 /.elasticbeanstalk/saved_configs/ 文件夹。

```
$ eb config save
```

以下示例说明如何使用 --cfg 选项将来自环境 tmp-dev 的配置设置保存到名为 v1-app-tmp.cfg.yml 的文件中。Elastic Beanstalk 将文件保存到文件夹 /.elasticbeanstalk/saved_configs/。如果不指定环境名称，Elastic Beanstalk 将保存来自当前运行的环境的配置设置。

```
$ eb config save tmp-dev --cfg v1-app-tmp
```

本部分提供了针对 eb config 命令（当此命令在没有子命令的情况下运行时）的示例。

以下命令在文本编辑器中显示当前环境的选项设置。

```
$ eb config
```

以下命令在文本编辑器中显示 my-env 环境的选项设置。

```
$ eb config my-env
```

以下示例显示了当前环境的选项设置。因为没有使用 --format 选项指定特定格式，它以 YAML 格式输出。

```
$ eb config --display
```

以下示例使用名为 example.txt 的文件中的规范更新当前环境的选项配置。此文件采用 YAML 或 JSON 格式。EB CLI 会自动检测文件格式。

- 命名空间 aws:autoscaling:asg 的 Minsize 选项设置为 1。
- 命名空间 aws:elasticbeanstalk:command 的批处理大小设置为 30%。
- 它从命名空间 AWSEBV2LoadBalancer.aws:elbv2:loadbalancer 删除了 IdleTimeout: None 选项设置。

```
$ eb config --update "file://example.txt"
```

Example - filename：**example.txt** – YAML 格式

```
OptionSettings:
  'aws:elasticbeanstalk:command':
    BatchSize: '30'
    BatchSizeType: Percentage
  'aws:autoscaling:asg':
    MinSize: '1'
OptionsToRemove:
  'AWSEBV2LoadBalancer.aws:elbv2:loadbalancer':
    IdleTimeout
```

Example - filename：**example.txt** – JSON 格式

```
{
    "OptionSettings": {
        "aws:elasticbeanstalk:command": {
            "BatchSize": "30",
            "BatchSizeType": "Percentage"
        },
        "aws:autoscaling:asg": {
            "MinSize": "1"
        }
    },
    "OptionsToRemove": {
        "AWSEBV2LoadBalancer.aws:elbv2:loadbalancer": {
            "IdleTimeout"
        }
    }
}
```

以下示例更新了当前环境的选项设置。该命令将命名空间 `aws:autoscaling:asg` 的 Minsize 选项设置为 1。

> **Note**
>
> 这些示例是特定于 Windows PowerShell 的。它们通过在双引号（"）字符前面加一个斜杠（\）字符来转义双引号字符。不同的操作系统和命令行环境可能具有不同的转义序列。因此，我们建议使用前面示例中显示的文件选项。在文件中指定配置选项不需要转义字符，并且在不同操作系统中一致。

下面是 JSON 格式示例。EB CLI 将检测格式是 JSON 还是 YAML。

```
PS C:\Users\myUser\EB_apps\myApp-env>eb config --update '{\"OptionSettings\":
{\"aws:autoscaling:asg\":{\"MaxSize\":\"1\"}}}'
```

下面是 YAML 格式示例。要以正确的格式输入 YAML 字符串，该命令需要包括 YAML 文件中所需的间距和行尾返回。

- 以“enter”或“Return”键结束每行。
- 第二行以两个空格开头，第三行以四个空格开头。

```
PS C:\Users\myUser\EB_apps\myApp-env>eb config --update 'OptionSettings:
>>  aws:autoscaling:asg:
>>    MinSize: \"1\"'
```

eb console

描述

打开浏览器可在 Elastic Beanstalk 管理控制台中显示环境配置控制面板。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令还会在 Elastic Beanstalk 管理控制台中显示 `platform.yaml` 中指定的生成器环境配置。

语法

eb console

eb console ***environment-name***

选项

名称	描述
常用选项	

eb create

描述

创建新环境并将应用程序版本部署到新环境。

> **Note**
>
> - 要对 .NET 应用程序使用 eb create，您必须按创建 .NET 应用程序的源包中所述创建部署程序包，然后按部署构件而不是项目文件夹中所述设置 CLI 配置以将程序包部署为构件。
> - 使用 EB CLI 创建环境需要服务角色。您可以通过在 Elastic Beanstalk 控制台中创建环境来创建服务角色。如果您没有服务角色，EB CLI 会尝试在您运行 `eb create` 时创建一个。

您可以从几个来源部署应用程序版本：

- 默认情况下：从本地项目目录中的应用程序源代码。
- 使用 `--version` 选项：从您的应用程序中已存在的应用程序版本。
- 当您的项目目录不具有应用程序代码时或使用 `--sample` 选项时：从环境平台特定的示例应用程序中部署。

语法

eb create

eb create ***environment-name***

环境名称长度必须在 4 到 40 个字符之间。名称只能包含字母、数字和连字符（-）。环境名称不能以连字符开头或结尾。

如果您在命令中包括了某个环境名称，EB CLI 不会提示您做出任何选择或创建服务角色。

如果您运行不带环境名称参数的命令，该命令将在交互流中运行，并提示您输入或选择一些设置的值。在此交互流中，如果您要部署示例应用程序，EB CLI 还会询问您是否要将此示例应用程序下载到本地项目目录。下载 EB CLI，您稍后便可以将 EB CLI 用于新环境，以运行需要应用程序代码（如 eb deploy）的操作。

某些交互式流提示仅在特定条件下显示。例如，如果您选择使用 Application Load Balancer，并且您的账户至少有一个可共享的 Application Load Balancer，则 Elastic Beanstalk 会显示一条提示，询问您

是否要使用共享的负载均衡器。如果您的账户中不存在可共享的 Application Load Balancer，则不会显示此提示。

Options

这些选项都不是必需的。如果您运行 eb create 时不指定任何选项，EB CLI 会提示您为每个设置输入或选择值。

名称	描述

`-d`
或
`--branch_default`
	将环境设置为当前存储库的默认环境。

`--cfg` *`config-name`*	使用保存的配置中的平台设置，保存的配置位于 `.elasticbeanstalk/saved_configs/` 或您的 Amazon S3 存储桶中。仅指定文件的名称，无需 `.cfg.yml` 扩展名。
`-c` *`subdomain-name`* 或 `--cname` *`subdomain-name`*	要为路由到您的网站的 CNAME DNS 条目添加前缀的子域名。 类型：字符串 默认值：环境名称

`-db`
或
`--database`
	将数据库附加到环境。如果您运行带 eb create 选项（而不带 `--database` 和 `--database.username` 选项）的 `--database.password`，EB CLI 将提示您输入主数据库的用户名和密码。

`-db.engine` *`engine`* 或 `--database.engine` *`engine`*	数据库引擎类型。如果您运行带此选项的 eb create，EB CLI 将启动附加了数据库的环境。即使您没有运行带 `--database` 选项的命令，亦是如此。 类型：字符串

名称	描述
	有效值：mysql、oracle-se1 、postgres、sqlserver-ex 、sqlserver-web、sqlserver-se
-db.i *instance_type* 或 --database.instance *instance_type*	要用于数据库的 Amazon EC2 实例的类型。如果您运行带此选项的 eb create，EB CLI 将启动附加了数据库的环境。即使您没有运行带 --database 选项的命令，亦是如此。 类型：字符串 有效值： Amazon RDS 支持一组标准的数据库实例。要为数据库引擎选择合适的数据库实例，您必须考虑一些特定的注意事项。有关更多信息，请参阅《Amazon RDS 用户指南》中的数据库实例类。
-db.pass *password* 或 --database.password *password*	数据库的密码。如果您运行带此选项的 eb create，EB CLI 将启动附加了数据库的环境。即使您没有运行带 --database 选项的命令，亦是如此。

名称	描述
-db.size *number_of_gigabytes* 或 --database.size *number_of_gigabytes*	要为数据库存储分配的 GB 数。如果您运行带此选项的 eb create，EB CLI 将启动附加了数据库的环境。即使您没有运行带 --database 选项的命令，亦是如此。 类型：数字 有效值： • MySQL – 5 到 1024。默认为 5。 • Postgres – 5 到 1024。默认为 5。 • Oracle – 10 到 1024。默认为 10。 • Microsoft SQL Server Express Edition – 30。 • Microsoft SQL Server Web Edition – 30。 • Microsoft SQL Server Standard Edition – 200。
-db.user *username* 或 --database.username *username*	数据库的用户名。如果您运行带此选项的 eb create，EB CLI 将启动附加了数据库的环境（即使您没有运行带 --database 选项的命令）。如果您运行带 eb create 选项（而不带 --database 和 --database.username 选项）的 --database.password ，EB CLI 将提示您输入主数据库用户名和密码。
-db.version *version* 或 --database.version *version*	可用于指定数据库引擎版本。如果此标志存在，环境将随带指定版本号的数据库一起启动（即使 --database 标志不存在）。
--elb-type *type*	负载均衡器类型。 类型：字符串 有效值：classic、application 、network 默认值：application

名称	描述

`-es`
 	
或
 	
`--enable-spot`
	为您的环境启用 Spot 实例请求。有关更多信息，请参阅Auto Scaling 组。
	
	相关选项：
	
	• `--instance-types`
	• `--on-demand-base-capacity`
	• `--on-demand-above-base-capacity`
	• `--spot-max-price`

`--env-group-suffix` *`groupname`*	要附加到环境名的组名。只能与编写环境一起使用。
`--envvars`	逗号分隔列表中的环境属性，格式为 *`name`*=*`value`*。有关限制，请参阅配置环境属性（环境变量）。
`-ip` *`profile_name`* 或 `--instance_profile` *`profile_name`*	具有 IAM 角色的实例配置文件，其中包含您的应用程序访问 AWS 资源所需的临时安全证书。

`-it`
 	
或者
 	
`--instance-types` *`type1`*`[,`*`type2`*` ...]`
	您希望环境使用的 Amazon EC2 实例类型的逗号分隔列表。如果不指定此选项，则 Elastic Beanstalk 将提供默认实例类型。
	
	有关更多信息，请参阅Amazon EC2 实例和Auto Scaling 组。
	
	Important
	EB CLI 仅将此选项应用于 Spot 实例。除非结合使用此选项和 `--enable-spot` 选项，EB CLI 才将忽略它。要为按需实例指定实例类型，请使用 `--intance-type`（无“s”）选项。

名称	描述

`-i`
 	
或
 	
`--instance_type`
	您希望环境使用的 Amazon EC2 实例类型。如果不指定此选项，则 Elastic Beanstalk 将提供默认实例类型。
	
	有关更多信息，请参阅Amazon EC2 实例。
	
	Important
	EB CLI 仅将此选项应用于按需实例。不要将此选项与 `--enable-spot` 选项一同使用，因为当您执行此操作时，EB CLI 会忽略它。要为 Spot 实例指定实例类型，请使用 `--intance-types`（无“s”）选项。

`-k` *key_name* 或 `--keyname` *key_name*	与 Secure Shell（SSH）客户端结合使用以安全登录运行 Elastic Beanstalk 应用程序的 Amazon EC2 实例的 Amazon EC2 密钥对的名称。如果您将此选项与 eb create 命令一起包含，则提供的值将覆盖您可能已使用 eb init 指定的任何密钥名称。 有效值：向 Amazon EC2 注册的现有密钥名称
`-im` *number-of-instances* 或者 `--min-instances` *number-of-instances*	您要求新环境拥有的最小 Amazon EC2 实例数。 类型：数字（整数） 默认值：1 有效值：1 到 10000
`-ix` *number-of-instances* 或者 `--max-instances` *number-of-instances*	您允许环境拥有的最大 Amazon EC2 实例数。 类型：数字（整数） 默认值：4 有效值：1 到 10000

名称	描述
`--modules` *component-a component-b*	要创建的组件环境的列表。只能与编写环境一起使用。

`-sb`
 	
或
 	
`--on-demand-base-capacity`
	扩展环境时，在考虑 Spot 实例之前，Auto Scaling 组预配置的最小按需实例数。
	
	此选项只能使用 `--enable-spot` 选项指定。有关更多信息，请参阅Auto Scaling 组。
	
	类型：数字（整数）
	
	默认值：`0`
	
	有效值：`0` 至 `--max-instances`（当不存在时：`MaxSize` ??? 命名空间中的 `aws:autoscaling:asg` 选项）

`-sp`
 	
或
 	
`--on-demand-above-base-capacity`
	Auto Scaling 组在 `--on-demand-base-capacity` 选项指定的超过的实例数作为额外容量预配置的按需实例的百分比。
	
	此选项只能使用 `--enable-spot` 选项指定。有关更多信息，请参阅Auto Scaling 组。
	
	类型：数字（整数）
	
	默认值：`0` 用于单个实例环境；`70` 用于负载均衡环境
	
	有效值：`0` 到 `100`

名称	描述
-p *platform-version* 或 --platform *platform-version*	要使用的平台版本。您可以指定平台、平台和版本、平台分支、解决方案堆栈名称或解决方案堆栈 ARN。例如： • php、PHP、node.js – 指定平台的最新平台版本 • php-7.2、"PHP 7.2" – 推荐的（通常是最新的）PHP 7.2 平台版本 • "PHP 7.2 running on 64bit Amazon Linux" – 此平台分支中推荐的（通常是最新的）PHP 平台版本 • "64bit Amazon Linux 2017.09 v2.6.3 running PHP 7.1" – 此解决方案堆栈名称指定的 PHP 平台版本 • "arn:aws:elasticbeanstalk:us-east-2::platform/PHP 7.1 running on 64bit Amazon Linux/2.6.3" – 此解决方案堆栈 ARN 指定的 PHP 平台版本 使用 eb platform list 获取可用配置的列表。 如果您指定 --platform 选项，则它会覆盖 eb init 期间提供的值。
-pr 或 --process	预处理并验证源代码包中的环境清单和配置文件。通过验证配置文件，可以在将应用程序版本部署到环境之前发现问题。
-r *region* 或 --region *region*	您要部署应用程序的 AWS 区域。 有关可以为此选项指定的值列表，请参阅 AWS 一般参考中的 AWS Elastic Beanstalk 端点和配额。
--sample	将示例应用程序部署到新环境，而不是您的存储库中的代码。

名称	描述
`--scale` *number-of-instances*	与指定数量的实例一起启动
`--service-role` *servicerole*	将非默认服务角色分配到环境。 **Note** 不要输入 ARN。仅输入角色名称。Elastic Beanstalk 会用正确的值为角色名称添加前缀，以在内部创建生成的 ARN。
`-ls` *#####* 或 `--shared-lb` *#####*	将环境配置为使用共享的负载均衡器。在您的账户中提供可共享的负载均衡器的名称或 ARN - 您显式创建而不是由其他 Elastic Beanstalk 环境创建的 Application Load Balancer。有关更多信息，请参阅共享 Application Load Balancer。 参数示例： • `FrontEndLB` - 负载均衡器名称。 • `arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/FrontEndLB/0dbf78d8ad96abbc` – Application Load Balancer ARN。 您只能使用 `--elb-type application` 指定此选项。如果您指定了该选项但未指定 `--shared-lb` ，则 Elastic Beanstalk 为环境创建一个专用的负载均衡器。

名称	描述

`-lp ##`
 	
或
 	
`--shared-lb-port ##`
	此环境的共享负载均衡器的默认侦听器端口。Elastic Beanstalk 添加了一个侦听器规则，该规则将来自此侦听器的所有流量路由到默认环境进程。有关更多信息，请参阅共享 Application Load Balancer。
	
	类型：数字（整数）
	
	默认值：`80`
	
	有效值：表示共享的负载均衡器的侦听器端口的任何整数。

`--single`	创建具有单个 Amazon EC2 实例并且没有负载均衡器的环境。 **Warning** 单实例环境不适用于生产。如果实例在部署期间变得不稳定，或者 Elastic Beanstalk 在配置更新期间终止并重新启动实例，则您的应用程序可能会在一段时间内不可用。可将单实例环境用于开发、测试或暂存。使用负载均衡环境进行生产。

`-sm`
 	
或
 	
`--spot-max-price`
	您愿意为 Spot 实例支付的每单位小时的最高价（美元）。
	
	此选项只能使用 `--enable-spot` 选项指定。有关更多信息，请参阅Auto Scaling 组。
	
	类型：数字（浮点数）
	
	默认：每种实例类型的按需价格。在这种情况下，该选项的值为 `null`。
	
	有效值：`0.001` 到 `20.0`
	
	有关竞价型实例最高价格选项的建议，请参阅 Amazon EC2 用户指南中的竞价型实例定价历史记录。

名称	描述
`--tags key1=value1[,key2=va`	在环境中标记资源。将标签指定为逗号分隔的 key=value 对的列表。 有关更多信息，请参阅 标记环境。

`-t worker`
 	
或
 	
`--tier worker`
	创建一个工作线程环境。忽略此选项，以创建 Web 服务器环境。

`--timeout minutes`	命令超时之前的设定分钟数。
`--version version_label`	指定要部署到环境中的应用程序版本，而不是本地项目目录中的应用程序员代码。 类型：字符串 有效值：现有的应用程序版本标签
`--vpc`	为您的环境配置 VPC。当您包含此选项时，EB CLI 会提示您在启动环境之前输入所有必需的设置。
`--vpc.dbsubnets subnet1,subnet2`	在 VPC 中指定数据库实例的子网。在指定 `--vpc.id` 时是必需的。
`--vpc.ec2subnets subnet1,subnet2`	在 VPC 中指定 Amazon EC2 实例的子网。在指定 `--vpc.id` 时是必需的。
`--vpc.elbpublic`	在 VPC 的公有子网中启动 Elastic Load Balancing 负载均衡器。 您无法使用 `--tier worker` 或 `--single` 选项指定此选项。

名称	描述
`--vpc.elbsubnets` *subnet1,subnet2*	在 VPC 中指定 Elastic Load Balancing 负载均衡器的子网。 您无法使用 `--tier worker` 或 `--single` 选项指定此选项。
`--vpc.id` *ID*	在指定的 VPC 中启动您的环境。
`--vpc.publicip`	在您的 VPC 中的公有子网中启动 Amazon EC2 实例。 您无法使用 `--tier worker` 选项指定此选项。
`--vpc.securitygroups` *securitygroup1,securitygroup2*	指定安全组 ID。在指定 `--vpc.id` 时是必需的。
常用选项	

输出

如果成功，则命令将通过问题来提示您，然后返回创建操作的状态。如果在启动期间发生问题，则可以使用 eb events 操作获取更多详细信息。

如果您在应用程序中启用了 CodeBuild 支持，则会在生成代码 CodeBuild 时eb create显示来自的信息。有关 Elastic Beanstalk 中 CodeBuild 支持的信息，请参阅。通过 AWS CodeBuild 使用 EB CLI

示例

下面的示例以交互模式创建环境。

```
$ eb create
Enter Environment Name
(default is tmp-dev): ENTER
Enter DNS CNAME prefix
(default is tmp-dev): ENTER
Select a load balancer type
1) classic
2) application
3) network
```

```
(default is 2): ENTER
Environment details for: tmp-dev
  Application name: tmp
  Region: us-east-2
  Deployed Version: app-141029_145448
  Environment ID: e-um3yfrzq22
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev.elasticbeanstalk.com
  Updated: 2014-10-29 21:54:51.063000+00:00
Printing Status:
...
```

下面的示例还在交互模式下创建环境。在此示例中，您的项目目录没有应用程序代码。该命令将部署一个示例应用程序，然后将其下载到您的本地项目目录中。

```
$ eb create
Enter Environment Name
(default is tmp-dev): ENTER
Enter DNS CNAME prefix
(default is tmp-dev): ENTER
Select a load balancer type
1) classic
2) application
3) network
(default is 2): ENTER
NOTE: The current directory does not contain any source code. Elastic Beanstalk is
 launching the sample application instead.
Do you want to download the sample application into the current directory?
(Y/n): ENTER
INFO: Downloading sample application to the current directory.
INFO: Download complete.
Environment details for: tmp-dev
  Application name: tmp
  Region: us-east-2
  Deployed Version: Sample Application
  Environment ID: e-um3yfrzq22
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev.elasticbeanstalk.com
  Updated: 2017-11-08 21:54:51.063000+00:00
Printing Status:
...
```

下面的命令创建一个环境但不显示任何提示。

```
$ eb create dev-env
Creating application version archive "app-160312_014028".
Uploading test/app-160312_014028.zip to S3. This may take a while.
Upload Complete.
Application test has been created.
Environment details for: dev-env
  Application name: test
  Region: us-east-2
  Deployed Version: app-160312_014028
  Environment ID: e-6fgpkjxyyi
  Platform: 64bit Amazon Linux 2015.09 v2.0.8 running PHP 5.6
  Tier: WebServer-Standard
  CNAME: UNKNOWN
  Updated: 2016-03-12 01:40:33.614000+00:00
Printing Status:
...
```

下面的命令在自定义 VPC 中创建一个环境。

```
$ eb create dev-vpc --vpc.id vpc-0ce8dd99 --vpc.elbsubnets subnet-
b356d7c6,subnet-02f74b0c --vpc.ec2subnets subnet-0bb7f0cd,subnet-3b6697c1 --
vpc.securitygroup sg-70cff265
Creating application version archive "app-160312_014309".
Uploading test/app-160312_014309.zip to S3. This may take a while.
Upload Complete.
Environment details for: dev-vpc
  Application name: test
  Region: us-east-2
  Deployed Version: app-160312_014309
  Environment ID: e-pqkcip3mns
  Platform: 64bit Amazon Linux 2015.09 v2.0.8 running Java 8
  Tier: WebServer-Standard
  CNAME: UNKNOWN
  Updated: 2016-03-12 01:43:14.057000+00:00
Printing Status:
...
```

eb deploy

描述

将应用程序源包从初始化的项目目录部署到正在运行的应用程序。

如果已安装 Git，EB CLI 将使用 `git archive` 命令根据最新的 `git commit` 命令的内容创建一个 `.zip` 文件。

但是，如果 `.ebignore` 存在于项目目录中，EB CLI 将不使用 git 命令和语义创建源包。这就意味着 EB CLI 将忽略 `.ebignore` 中指定的文件，并包括所有其他文件。具体而言，它会包括未提交的源文件。

> **Note**
>
> 您可以将 EB CLI 配置为从构建过程部署工件而不是创建项目文件夹 ZIP 文件。有关更多信息，请参阅部署构件而不是项目文件夹。

语法

eb deploy

eb deploy ***environment-name***

选项

名称	描述
`-l` *`version_label`* 或 `--label` *`version_label`*	指定要用作 EB CLI 所创建版本的标签。如果该标签已被使用，EB CLI 将重新部署使用该标签的先前版本。 类型：字符串
`--env-group-suffix` *`groupname`*	要附加到环境名的组名。只能与编写环境一起使用。
`-m "`*`version_description`*`"`	应用程序版本的描述（用双引号引起来）。

名称	描述
或 `--message "version_description "`	类型：字符串
`--modules component-a component-b`	要更新的组件的列表。只能与编写环境一起使用。

`-p`
 	
或
 	
`--process`
	预处理并验证源代码包中的环境清单和配置文件。通过验证配置文件，可以在将应用程序版本部署到环境之前发现问题。

`--source codecommit/ repository-name/branch-name`	CodeCommit 存储库和分支。请参阅通过 AWS CodeCommit 使用 EB CLI。
`--staged`	部署暂存在 Git 索引中而不是 HEAD 提交中的文件。
`--timeout minutes`	命令超时之前的分钟数。
`--version version_label`	要部署的现有应用程序版本。 类型：字符串
常用选项	

输出

如果成功，则该命令返回 `deploy` 操作的状态。

如果您在应用程序中启用了 CodeBuild 支持，则 eb deploy 会在生成代码时显示 CodeBuild 中的信息。有关 Elastic Beanstalk 中的 CodeBuild 支持的信息，请参阅通过 AWS CodeBuild 使用 EB CLI。

示例

下面的示例将部署当前应用程序。

```
$ eb deploy
2018-07-11 21:05:22    INFO: Environment update is starting.
2018-07-11 21:05:27    INFO: Deploying new version to instance(s).
2018-07-11 21:05:53    INFO: New application version was deployed to running EC2
 instances.
2018-07-11 21:05:53    INFO: Environment update completed successfully.
```

eb events

描述

返回环境的最近事件。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则此命令还会返回构建器环境的最新事件。

语法

eb events

eb events ***environment-name***

选项

名称	描述

`-f`
或
`--follow`
	流式传输事件。要取消，请按 Ctrl+C。

输出

如果成功，命令将返回最新事件。

示例

下面的示例返回 个最近事件。

```
$ eb events
```

```
2014-10-29 21:55:39     INFO    createEnvironment is starting.
2014-10-29 21:55:40     INFO    Using elasticbeanstalk-us-west-2-111122223333 as Amazon
 S3 storage bucket for environment data.
2014-10-29 21:55:57     INFO    Created load balancer named: awseb-e-r-AWSEBLoa-
NSKUOK5X6Z9J
2014-10-29 21:56:16     INFO    Created security group named: awseb-e-rxgrhjr9bx-stack-
AWSEBSecurityGroup-1UUHU5LZ20ZY7
2014-10-29 21:57:18     INFO    Waiting for EC2 instances to launch. This may take a
 few minutes.
2014-10-29 21:57:18     INFO    Created Auto Scaling group named: awseb-e-rxgrhjr9bx-
stack-AWSEBAutoScalingGroup-1TE320ZCJ9RPD
2014-10-29 21:57:22     INFO    Created Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:2cced9e6-859b-421a-
be63-8ab34771155a:autoScalingGroupName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingGroup-1TE320ZCJ9RPD:policyName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingScaleUpPolicy-1I2ZSNVU4APRY
2014-10-29 21:57:22     INFO    Created Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:1f08b863-
bf65-415a-b584-b7fa3a69a0d5:autoScalingGroupName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingGroup-1TE320ZCJ9RPD:policyName/awseb-e-rxgrhjr9bx-stack-
AWSEBAutoScalingScaleDownPolicy-1E3G7PZKZPSOG
2014-10-29 21:57:25     INFO    Created CloudWatch alarm named: awseb-e-rxgrhjr9bx-
stack-AWSEBCloudwatchAlarmLow-VF5EJ549FZBL
2014-10-29 21:57:25     INFO    Created CloudWatch alarm named: awseb-e-rxgrhjr9bx-
stack-AWSEBCloudwatchAlarmHigh-LA9YEW3O6WJO
2014-10-29 21:58:50     INFO    Added EC2 instance 'i-c7ee492d' to Auto ScalingGroup
 'awseb-e-rxgrhjr9bx-stack-AWSEBAutoScalingGroup-1TE320ZCJ9RPD'.
2014-10-29 21:58:53     INFO    Successfully launched environment: tmp-dev
2014-10-29 21:59:14     INFO    Environment health has been set to GREEN
2014-10-29 21:59:43     INFO    Adding instance 'i-c7ee492d' to your environment.
```

eb health

描述

返回环境的最新运行状况。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令还会返回生成器环境的最新运行状况。

语法

eb health

eb health ***environment-name***

选项

名称	描述
-r 或 --refresh	以交互方式显示运行状况信息并在报告新信息时每 10 秒更新一次。
--mono	不要在输出中显示颜色。

输出

如果成功，命令将返回最新的运行状况。

示例

以下示例返回 Linux 环境的最新运行状况信息。

```
~/project $ eb health
 elasticBeanstalkExa-env                                Ok
 2015-07-08 23:13:20
WebServer
 Ruby 2.1 (Puma)
  total      ok    warning  degraded  severe    info   pending  unknown
    5        5        0        0        0        0        0        0

  instance-id   status     cause
                                        health
    Overall     Ok
  i-d581497d    Ok
  i-d481497c    Ok
  i-136e00c0    Ok
  i-126e00c1    Ok
  i-8b2cf575    Ok

  instance-id   r/sec    %2xx   %3xx   %4xx   %5xx      p99      p90      p75     p50
   p10                                requests
```

```
    Overall     671.8   100.0    0.0    0.0    0.0    0.003    0.002    0.001   0.001
0.000
 i-d581497d    143.0    1430      0      0      0    0.003    0.002    0.001   0.001
0.000
 i-d481497c    128.8    1288      0      0      0    0.003    0.002    0.001   0.001
0.000
 i-136e00c0    125.4    1254      0      0      0    0.004    0.002    0.001   0.001
0.000
 i-126e00c1    133.4    1334      0      0      0    0.003    0.002    0.001   0.001
0.000
 i-8b2cf575    141.2    1412      0      0      0    0.003    0.002    0.001   0.001
0.000

 instance-id   type       az   running     load 1  load 5      user%  nice%  system%
idle%   iowait%                              cpu
 i-d581497d    t2.micro   1a   12 mins       0.0    0.04        6.2    0.0      1.0
92.5       0.1
 i-d481497c    t2.micro   1a   12 mins      0.01    0.09        5.9    0.0      1.6
92.4       0.1
 i-136e00c0    t2.micro   1b   12 mins      0.15    0.07        5.5    0.0      0.9
93.2       0.0
 i-126e00c1    t2.micro   1b   12 mins      0.17    0.14        5.7    0.0      1.4
92.7       0.1
 i-8b2cf575    t2.micro   1c   1 hour       0.19    0.08        6.5    0.0      1.2
92.1       0.1

 instance-id   status     id   version              ago
                                 deployments
 i-d581497d    Deployed   1    Sample Application   12 mins
 i-d481497c    Deployed   1    Sample Application   12 mins
 i-136e00c0    Deployed   1    Sample Application   12 mins
 i-126e00c1    Deployed   1    Sample Application   12 mins
 i-8b2cf575    Deployed   1    Sample Application   1 hour
```

eb init

描述

通过一系列问题来提示您，设置使用 EB CLI 创建的 Elastic Beanstalk 应用程序的默认值。

> **Note**
>
> 使用 eb init 设置的值仅应用于当前计算机上的当前目录和存储库。

该命令不会在您的 Elastic Beanstalk 账户中创建任何内容。要创建 Elastic Beanstalk 环境，请在运行 eb init 后运行 eb create。

语法

eb init

eb init *application-name*

选项

如果您运行 eb init 时不指定 `--platform` 选项，EB CLI 会提示您为每个设置输入值。

> **Note**
>
> 要使用 eb init 创建新的密钥对，`ssh-keygen` 必须已在本地计算机上安装且能够从命令行访问。

名称	描述	
`-i` `--interactive`	强制 EB CLI 提示您为每个 eb init 命令选项提供值。 **Note** `init` 命令提示您为不具有（默认）值的 eb init 命令选项提供值。在您首次在目录中运行 eb init 命令后，EB CLI 可能不会提示您输入任何命令选项。因此，当您想更改之前设定的设置时，请使用 `--interactive` 选项。	
`-k` *keyname* `--keyname` *keyname*	与安全外壳（SSH）客户端结合使用以安全登录运行 Elastic Beanstalk 应用程序的 Amazon EC2 实例的 Amazon EC2 密钥对的名称。	
`--modules` *folder-1 folder-2*	要初始化的子目录的列表。只能与编写环境一起使用。	

名称	描述	
-p *platform-version* --platform *platform-version*	要使用的平台版本。您可以指定平台、平台和版本、平台分支、解决方案堆栈名称或解决方案堆栈 ARN。例如： • php、PHP、node.js – 指定平台的最新平台版本 • php-7.2、"PHP 7.2" – 推荐的（通常是最新的）PHP 7.2 平台版本 • "PHP 7.2 running on 64bit Amazon Linux" – 此平台分支中推荐的（通常是最新的）PHP 平台版本 • "64bit Amazon Linux 2017.09 v2.6.3 running PHP 7.1" – 此解决方案堆栈名称指定的 PHP 平台版本 • "arn:aws:elasticbeanstalk:us-east-2::platform/PHP 7.1 running on 64bit Amazon Linux/2.6.3" – 此解决方案堆栈 ARN 指定的 PHP 平台版本 使用 eb platform list 获取可用配置的列表。 指定 --platform 选项可跳过交互式配置。 **Note** 在指定此选项时，EB CLI 不会提示您输入任何其他选项的值。相反，它将假定每个选项的默认值。您可以为不需要为其使用默认值的任何项指定选项。	
--source codecommit/ *repository-name*/*branch-name*	CodeCommit 存储库和分支。请参阅通过 AWS CodeCommit 使用 EB CLI。	

名称	描述	
--tags *key1*=*value1*	标记应用程序。将标签指定为逗号分隔的 key=value 对的列表。 有关更多信息，请参阅标记应用程序。	
常用选项		

CodeBuild 支持

如果您在包含 buildspec.yml 文件的文件夹中运行 eb init，Elastic Beanstalk 会使用特定于 Elastic Beanstalk 的选项解析文件中的 eb_codebuild_settings 条目。有关 Elastic Beanstalk 中的 CodeBuild 支持的信息，请参阅通过 AWS CodeBuild 使用 EB CLI。

输出

如果成功，该命令将通过一系列提示指导您设置新的 Elastic Beanstalk 应用程序。

示例

下面的示例请求对 EB CLI 进行初始化，并提示您输入有关您的应用程序的信息。将###文本替换为您自己的值。

```
$ eb init -i
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 3

Select an application to use
1) HelloWorldApp
2) NewApp
3) [ Create new Application ]
```

```
(default is 3): 3

Enter Application Name
(default is "tmp"):
Application tmp has been created.

It appears you are using PHP. Is this correct?
(y/n): y

Select a platform branch.
1) PHP 7.2 running on 64bit Amazon Linux
2) PHP 7.1 running on 64bit Amazon Linux (Deprecated)
3) PHP 7.0 running on 64bit Amazon Linux (Deprecated)
4) PHP 5.6 running on 64bit Amazon Linux (Deprecated)
5) PHP 5.5 running on 64bit Amazon Linux (Deprecated)
6) PHP 5.4 running on 64bit Amazon Linux (Deprecated)
(default is 1): 1
Do you want to set up SSH for your instances?
(y/n): y

Select a keypair.
1) aws-eb
2) [ Create new KeyPair ]
(default is 2): 1
```

eb labs

描述

eb labs 的子命令支持在制品功能或试验功能。在将来版本的 EB CLI 中，这些命令可能被删除或改动，且不一定向前兼容。

有关可用子命令和说明的列表，请运行 eb labs --help。

eb list

描述

按照 `--all` 选项所指定的方式列出当前应用程序中的所有环境或所有应用程序中的所有环境。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令还会列出生成器环境。

语法

eb list

选项

名称	描述
-a 或 --all	列出来自所有应用程序的所有环境。
-v 或 --verbose	提供有关所有环境（包括实例）的更多详细信息。
常用选项	

输出

如果成功，命令将返回环境名称列表，其中您的当前环境用星号（*）标记。

示例 1

以下示例列出您的环境，并指示 tmp-dev 是您的默认环境。

```
$ eb list
* tmp-dev
```

示例 2

以下示例列出您的环境和附加详细信息。

```
$ eb list --verbose
Region: us-west-2
Application: tmp
    Environments: 1
```

```
        * tmp-dev : ['i-c7ee492d']
```

eb local

描述

使用 eb local run 可在 Docker 本地运行您的应用程序容器。使用 eb local status 检查应用程序容器的状态。使用 eb local open 在 Web 浏览器中打开应用程序。使用 eb local logs 检索应用程序日志的位置。

使用 eb local setenv 和 eb local printenv 可以设置和查看环境变量，这些环境变量是为您使用 eb local run 本地运行的 Docker 容器提供的。

必须在 Docker 应用程序项目目录中运行所有 eb local 命令，该应用程序已使用 eb init 初始化为 EB CLI 存储库。

> **Note**
>
> 在运行 Linux 或 macOS 的本地计算机上使用 eb local。该命令不支持 Windows。
> 在 macOS 上使用此命令之前，请安装 Docker for Mac，并确保未安装 boot2docker（或其不在执行路径中）。eb local 命令会尝试使用 boot2docker（如果它存在），但两者在 macOS 上不能很好地配合使用。

语法

eb local run

eb local status

eb local open

eb local logs

eb local setenv

eb local printenv

选项

eb local run

名称	描述
`--envvars key1=value1,key2=value2`	设置 EB CLI 将传递到本地 Docker 容器的环境变量。在多容器环境中，所有变量将传递到所有容器。
`--port hostport`	将主机上的端口映射到容器上公开的端口。如果您不指定此选项，EB CLI 将在主机和容器上使用相同的端口。 此选项仅适用于 Docker 平台应用程序。它不适用于多容器 Docker 平台。
常用选项	

eb local status

eb local open

eb local logs

eb local setenv

eb local printenv

名称	描述
常用选项	

输出

eb local run

来自 Docker 的状态消息。只要应用程序在运行，就保持有效。按 Ctrl+C 可停止应用程序。

eb local status

应用程序使用的每个容器的状态，即是否正在运行。

eb local open

在 Web 浏览器中打开应用程序并退出。

eb local logs

由使用 eb local run 本地运行的应用程序在项目目录中生成日志的位置。

eb local setenv

无

eb local printenv

使用 eb local setenv 设置的环境变量的名称和值。

示例

eb local run

```
~/project$ eb local run
Creating elasticbeanstalk_phpapp_1...
Creating elasticbeanstalk_nginxproxy_1...
Attaching to elasticbeanstalk_phpapp_1, elasticbeanstalk_nginxproxy_1
phpapp_1     | [23-Apr-2015 23:24:25] NOTICE: fpm is running, pid 1
phpapp_1     | [23-Apr-2015 23:24:25] NOTICE: ready to handle connections
```

eb local status

查看本地容器的状态：

```
~/project$ eb local status
Platform: 64bit Amazon Linux 2014.09 v1.2.1 running Multi-container Docker 1.3.3
 (Generic)
Container name: elasticbeanstalk_nginxproxy_1
Container ip: 127.0.0.1
Container running: True
Exposed host port(s): 80
Full local URL(s): 127.0.0.1:80

Container name: elasticbeanstalk_phpapp_1
Container ip: 127.0.0.1
Container running: True
Exposed host port(s): None
```

```
Full local URL(s): None
```

eb local logs

查看当前项目的日志路径：

```
~/project$ eb local logs
Elastic Beanstalk will write logs locally to /home/user/project/.elasticbeanstalk/logs/
local.
Logs were most recently created 3 minutes ago and written to /home/user/
project/.elasticbeanstalk/logs/local/150420_234011665784.
```

eb local setenv

设置要用于 eb local run 的环境变量。

```
~/project$ eb local setenv PARAM1=value
```

输出使用 eb local setenv 设置的环境变量。

```
~/project$ eb local printenv
Environment Variables:
PARAM1=value
```

eb logs

描述

eb logs 命令具有两种不同的用途：启用或禁用至 CloudWatch Logs 的日志流式传输和检索实例日志或 CloudWatch Logs 日志。此命令与 `--cloudwatch-logs`（`-cw`）选项结合使用时启用或禁用日志流式传输。如果没有此选项，则它将检索日志。

检索日志时，请指定 `--all`、`--zip` 或 `--stream` 选项以检索全部日志。如果未指定上述任何选项，则 Elastic Beanstalk 将检索结尾日志。

此命令将处理指定或默认环境的日志。相关日志因容器类型而异。如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则此命令还会处理生成器环境的日志。

有关更多信息，请参阅the section called “CloudWatch Logs”。

语法

启用或禁用到 CloudWatch Logs 的日志流式传输：

```
eb logs --cloudwatch-logs [enable | disable] [--cloudwatch-log-source instance |
 environment-health | all] [environment-name]
```

检索实例日志：

```
eb logs [-all | --zip | --stream] [--cloudwatch-log-source instance] [--
instance instance-id] [--log-group log-group] [environment-name]
```

检索环境运行状况日志：

```
eb logs [-all | --zip | --stream] --cloudwatch-log-source environment-health
 [environment-name]
```

选项

名称	描述

`-cw [enable | disable]`
 	
或
 	
`--cloudwatch-logs [enable | disable]`
	启用或禁用到 CloudWatch Logs 的日志流式传输。如果未提供任何参数，则将启用日志流式传输。此外，如果未指定 `--cloudwatch-log-source` （`-cls`）选项，则将启用或禁用实例日志流式传输。

`-cls instance | environment-health | all`
 	
或
 	
`--cloudwatch-log-source instance | environment-health | all`
	与 CloudWatch Logs 结合使用时，请指定日志的源。使用此命令的启用或禁用形式时，这些是要为之启用或禁用 CloudWatch Logs 流式传输的日志。使用此命令的检索形式时，这些是要从 CloudWatch Logs 中检索的日志。
	
	有效值：
	
	• 有 `--cloudwatch-logs`（启用或禁用）– `instance | environment-health | all`
	• 没有 `--cloudwatch-logs`（检索）– `instance | environment-health`

名称	描述
	值含义： • instance（默认）– 实例日志 • environment-health – 环境运行状况日志（当环境中启用增强型运行状况时受支持） • all – 两种日志源
-a 或 --all	检索全部日志并将它们保存到 .elasticbeanstalk/logs 目录中。
-z 或 --zip	检索全部日志，将它们压缩为一个 .zip 文件，然后将该文件保存到 .elasticbeanstalk/logs 目录中。
--stream	流式传输（持续输出）全部日志。与此选项结合使用时，此命令在中断之前将一直运行（按 **Ctrl+C**）。
-i *instance-id* 或 --instance *instance-id*	仅检索指定实例的日志。

名称	描述

`-g log-group`
或
`--log-group log-group`
	指定要从中检索日志的 CloudWatch Logs 日志组。此选项仅当启用至 CloudWatch Logs 的实例日志流式传输时有效。
	如果启用了实例日志流式传输，并且未指定 `--log-group` 选项，则默认日志组为以下项之一：
	• Amazon Linux 2 – `/aws/elasticbeanstalk/ environment-name /var/log/eb-engine.log`
	• Windows 平台 – `/aws/elasticbeanstalk/ environment-name /EBDeploy-Log`
	• Amazon Linux AMI (AL1) – `/aws/elasticbeanstalk/ environment-name /var/log/eb-activity.log`
	Note
	2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。有关迁移到当前且完全受支持的 Amazon Linux 2023 平台分支的更多信息，请参阅 将 Elastic Beanstalk Linux 应用程序迁移到 Amazon Linux 2023 或 Amazon Linux 2。
	有关每个日志文件对应的日志组的信息，请参阅Elastic Beanstalk 如何设置 CloudWatch Logs。

常用选项	

输出

默认情况下，直接在终端显示日志。使用分页程序显示输出。按 **Q** 或 **q** 退出。

使用 `--stream` 时，在终端显示现有日志并保持运行。按 **Ctrl+C** 退出。

使用 `--all` 和 `--zip` 时，将日志保存到本地文件并显示文件位置。

示例

以下示例启用至 CloudWatch Logs 的实例日志流式传输。

```
$ eb logs -cw enable
Enabling instance log streaming to CloudWatch for your environment
After the environment is updated you can view your logs by following the link:
https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#logs:prefix=/aws/
elasticbeanstalk/environment-name/
Printing Status:
2018-07-11 21:05:20    INFO: Environment update is starting.
2018-07-11 21:05:27    INFO: Updating environment environment-name's configuration
 settings.
2018-07-11 21:06:45    INFO: Successfully deployed new configuration to environment.
```

以下示例将实例日志检索到 .zip 文件中。

```
$ eb logs --zip
Retrieving logs...
Logs were saved to /home/workspace/environment/.elasticbeanstalk/logs/150622_173444.zip
```

eb open

描述

在默认浏览器中打开网站的公共 URL。

语法

eb open

eb open ***environment-name***

选项

名称	描述
常用选项	

输出

命令 eb open 不具有任何输出。相反，它会在浏览器窗口中打开应用程序。

eb platform

描述

此命令支持两种不同工作区：

平台

使用此命令可管理自定义平台。

环境 ()

使用此工作区可选择默认平台或显示有关当前平台的信息。

Elastic Beanstalk 提供了 eb platform 的快捷方式 ebp。

> **Note**
>
> Windows PowerShell 使用 ebp 作为命令别名。如果您在 Windows PowerShell 中运行 EB CLI，请使用此命令的长格式 – eb platform。

对自定义平台使用 eb platform

列出当前平台的版本并能让您管理自定义平台。

语法

eb platform create [***version***] [***options***]

eb platform delete [***version***] [***options***]

eb platform events [***version***] [***options***]

eb platform init [***platform***] [***options***]

eb platform list [***options***]

eb platform logs [***version***] [***options***]

eb platform status [***version***] [***options***]

eb platform use [***platform***] [***options***]

选项

名称	描述
`create [version] [options]`	生成平台的新版本。了解更多。
`delete version [options]`	删除平台版本。了解更多。
`events [version] [options]`	显示平台版本中的事件。了解更多。
`init [platform] [options]`	初始化平台存储库。了解更多。
`list [options]`	列出当前平台的版本。了解更多。
`logs [version] [options]`	显示平台版本的生成器环境中的日志。了解更多。
`status [version] [options]`	显示平台版本的状态。了解更多。
`use [platform] [options]`	选择生成新版本所基于的其他平台。了解更多。
常用选项	

常用选项

所有 eb platform 命令均包含以下常用选项。

名称	描述
`-h`	显示帮助消息并退出。

名称	描述
OR `--help`	
`--debug`	显示更多调试输出。
`--quiet`	隐藏所有输出。

`-v`
 	
OR
 	
`--verbose`
	显示更多输出。

`--profile` *PROFILE*	使用您凭证中的指定 *PROFILE*。
`-r` *REGION* OR `--region` *REGION*	使用区域 *REGION*。
`--no-verify-ssl`	请勿验证 AWS SSL 证书。

Eb platform create

生成平台的新版本并返回新版本的 ARN。如果当前区域没有正在运行的生成器环境，则此命令将会启动一个。*version* 和递增选项（`-M`、`-m` 和 `-p`）是互斥的。

选项

名称	描述
##	如果未指定 *version*，请基于最新平台创建新版本，且修补版本（n.n.N 中的 N）递增一。
`-M` OR	将主要版本号（N.n.n 中的 N）递增一。

名称	描述
`--major-increment`	

`-m`
 	
OR
 	
`--minor-increment`
	将次要版本号（n.N.n 中的 N）递增一。

`-p`
 	
OR
 	
`--patch-increment`
	将修补版本号（n.n.N 中的 N）递增一。

`-i` *INSTANCE_TYPE* OR --instance-type *INSTANCE_TYPE*	使用 *INSTANCE_TYPE* 作为实例类型，例如 **t1.micro**。
`-ip` *INSTANCE_PROFILE* OR `--instance-profile` *INSTANCE_PROFILE*	在为自定义平台创建 AMI 时，将 *INSTANCE_PROFILE* 用作实例配置文件。 如果未指定 `-ip` 选项，请创建实例配置文件 `aws-elasticbeanstalk-custom-platforme-ec2-role` 并将它用于自定义平台。
`--tags` *key1*=*value1*[,*ke*	标记自定义平台版本。将标签指定为逗号分隔的 key=value 对的列表。 有关更多信息，请参阅标记自定义平台版本。
`--timeout` *minutes*	命令超时之前的设定分钟数。
`--vpc.id` *VPC_ID*	在其中生成 Packer 的 VPC 的 ID。
`--vpc.subnets` *VPC_SUBNETS*	在其中生成 Packer 的 VPC 子网。

名称	描述
`--vpc.publicip`	将公有 IP 与启动的 EC2 实例关联。

Eb platform delete

删除平台版本。如果环境在使用该版本，则该版本不会被删除。

选项

名称	描述
`version`	要删除的版本。此值为必填项。
`--cleanup`	删除所有处于 `Failed` 状态的平台版本。
`--all-platforms`	如果指定了 `--cleanup` ，请删除所有平台的处于 `Failed` 状态的所有平台版本。
`--force`	在删除版本时，不要求进行确认。

Eb platform events

显示平台版本中的事件。如果指定了 *`version`*，则显示该版本中的事件，否则显示当前版本中的事件。

选项

名称	描述
`##`	为其显示事件的版本。此值为必填项。

`-f`
 	
OR
 	
`--follow`
	继续在事件发生时显示它们。

Eb platform init

初始化平台存储库。

选项

名称	描述
platform	要初始化的平台的名称。该值是必需的，除非启用了 -i（交互式模式）。
-i OR --interactive	使用交互式模式。
-k *KEYNAME* OR --keyname *KEYNAME*	默认 EC2 密钥名称。

您可以在之前初始化的目录中运行该命令，但是，如果是在之前初始化的目录中运行，则无法更改工作区类型。

要使用不同选项重新初始化，请使用 -i 选项。

Eb platform list

列出与工作区（目录）或区域关联的平台的版本。

该命令将返回不同的结果，具体取决于您在其中运行它的工作区的类型，如下所示：

- 在平台工作区（由 eb platform init 初始化的目录）中，该命令将返回工作区中定义的自定义平台的所有平台版本的列表。添加 --all-platforms 或 --verbose 选项以获取您的账户在与工作区关联的区域中具有的所有自定义平台的所有平台版本的列表。
- 在应用程序工作区（由 eb init 初始化的目录）中，该命令将返回由 Elastic Beanstalk 托管的平台和您账户的自定义平台的所有平台版本的列表。列表使用简短的平台版本名称，而且一些平台版本变体可进行组合。添加 --verbose 选项以获取包含完整名称及分别列出的所有变体的详细列表。

- 在未初始化的目录中，该命令仅适用于 `--region` 选项。它将返回区域中支持的所有 Elastic Beanstalk 托管的平台版本的列表。列表使用简短的平台版本名称，而且一些平台版本变体可进行组合。添加 `--verbose` 选项以获取包含完整名称及分别列出的所有变体的详细列表。

选项

名称	描述

`-a`
OR
`--all-platforms`
	仅在已初始化的工作区（由 `eb platform init` 或 `eb init` 初始化的目录）中有效。列出与您的账户关联的所有自定义平台的平台版本。

`-s` *`STATUS`* OR `--status` *`STATUS`*	仅列出与 *`STATUS`* 匹配的平台： • Ready • 已失败 • 删除 • 创建

Eb platform logs

显示平台版本的生成器环境中的日志。

选项

名称	描述
`version`	为其显示日志的平台的版本。如果省略，则显示当前版本的日志。
`--stream`	流式传输使用 CloudWatch 设置的部署日志。

Eb platform status

显示平台版本的状态。

选项

名称	描述
version	为其检索状态的平台的版本。如果省略，则显示当前版本的状态。

Eb platform use

选择生成新版本所基于的其他平台。

选项

名称	描述
platform	将 *platform* 指定为该工作区的活动版本。此值为必填项。

将 eb platform 用于环境

列出支持的平台，并使您能够设置默认平台和在启动环境时要使用的平台版本。使用 eb platform list 可查看所有支持平台的列表。使用 eb platform select 可更改项目的平台。使用 eb platform show 可查看您的项目的选定平台。

语法

eb platform list

eb platform select

eb platform show

选项

名称	描述
`list`	列出当前平台的版本。
`select`	选择默认平台。

名称	描述
show	显示有关当前平台的信息。

示例 1

以下示例列出 Elastic Beanstalk 支持的所有平台的所有配置的名称。

```
$ eb platform list
docker-1.5.0
glassfish-4.0-java-7-(preconfigured-docker)
glassfish-4.1-java-8-(preconfigured-docker)
go-1.3-(preconfigured-docker)
go-1.4-(preconfigured-docker)
iis-7.5
iis-8
iis-8.5
multi-container-docker-1.3.3-(generic)
node.js
php-5.3
php-5.4
php-5.5
python
python-2.7
python-3.4
python-3.4-(preconfigured-docker)
ruby-1.9.3
ruby-2.0-(passenger-standalone)
ruby-2.0-(puma)
ruby-2.1-(passenger-standalone)
ruby-2.1-(puma)
ruby-2.2-(passenger-standalone)
ruby-2.2-(puma)
tomcat-6
tomcat-7
tomcat-7-java-6
tomcat-7-java-7
tomcat-8-java-8
```

示例 2

以下示例提示您从平台列表中选择并选择要为指定平台部署的版本。

```
$ eb platform select
Select a platform.
1) PHP
2) Node.js
3) IIS
4) Tomcat
5) Python
6) Ruby
7) Docker
8) Multi-container Docker
9) GlassFish
10) Go
(default is 1): 5

Select a platform version.
1) Python 2.7
2) Python
3) Python 3.4 (Preconfigured - Docker)
```

示例 3

以下示例显示有关当前默认平台的信息。

```
$ eb platform show
Current default platform: Python 2.7
New environments will be running:  64bit Amazon Linux 2014.09 v1.2.0 running Python 2.7

Platform info for environment "tmp-dev":
Current: 64bit Amazon Linux 2014.09 v1.2.0 running Python
Latest:  64bit Amazon Linux 2014.09 v1.2.0 running Python
```

eb printenv

描述

在命令窗口中打印所有环境属性。

语法

eb printenv

eb printenv ***environment-name***

选项

名称	描述
常用选项	

输出

如果成功，则该命令返回 printenv 操作的状态。

示例

以下示例将打印指定环境的环境属性。

```
$ eb printenv
Environment Variables:
     PARAM1 = Value1
```

eb restore

描述

重建已终止的环境，创建具有相同名称、ID 和配置的新环境。环境名称、域名和应用程序版本必须可用才能重建成功。

语法

eb restore

eb restore ***environment_id***

选项

名称	描述
常用选项	

输出

EB CLI 显示可用于恢复的已终止环境的列表。

示例

```
$ eb restore
Select a terminated environment to restore

  #   Name          ID             Application Version      Date Terminated        Ago

  3   gamma         e-s7mimej8e9   app-77e3-161213_211138   2016/12/14 20:32 PST   13
 mins
  2   beta          e-sj28uu2wia   app-77e3-161213_211125   2016/12/14 20:32 PST   13
 mins
  1   alpha         e-gia8mphu6q   app-77e3-161213_211109   2016/12/14 16:21 PST   4
 hours

 (Commands: Quit, Restore, # #)

Selected environment alpha
Application:    scorekeep
Description:    Environment created from the EB CLI using "eb create"
CNAME:          alpha.h23tbtbm92.us-east-2.elasticbeanstalk.com
Version:        app-77e3-161213_211109
Platform:       64bit Amazon Linux 2016.03 v2.1.6 running Java 8
Terminated:     2016/12/14 16:21 PST
Restore this environment? [y/n]: y

2018-07-11 21:04:20    INFO: restoreEnvironment is starting.
2018-07-11 21:04:39    INFO: Created security group named: sg-e2443f72
...
```

eb scale

描述

通过将最小实例数和最大实例数设置为指定数字，扩展环境以始终在指定数量的实例上运行。

语法

eb scale ***number-of-instances***

eb scale ***number-of-instances environment-name***

选项

名称	描述
--timeout	命令超时之前的分钟数。
常用选项	

输出

如果成功，命令会将要运行的实例的最小数目和最大数目更新为指定数目。

示例

以下示例将实例数设置为 2。

```
$ eb scale 2
2018-07-11 21:05:22    INFO: Environment update is starting.
2018-07-11 21:05:27    INFO: Updating environment tmp-dev's configuration settings.
2018-07-11 21:08:53    INFO: Added EC2 instance 'i-5fce3d53' to Auto Scaling Group
 'awseb-e-2cpfjbra9a-stack-AWSEBAutoScalingGroup-7AXY7U13ZQ6E'.
2018-07-11 21:08:58    INFO: Successfully deployed new configuration to environment.
2018-07-11 21:08:59    INFO: Environment update completed successfully.
```

eb setenv

描述

设置默认环境的环境属性。

语法

eb setenv ***key*=*value***

您可以按需添加任意数目的属性，但是所有属性的总大小不得超过 4096 字节。您可以通过将值留空来删除变量。有关限制，请参阅配置环境属性（环境变量）。

> **Note**
>
> 如果 value 包含特殊字符，则必须使用前置 \ 字符将该字符转义。

选项

名称	描述
--timeout	命令超时之前的分钟数。
常用选项	

输出

如果成功，命令将显示环境更新成功。

示例

以下示例设置环境变量 ExampleVar。

```
$ eb setenv ExampleVar=ExampleValue
2018-07-11 21:05:25    INFO: Environment update is starting.
2018-07-11 21:05:29    INFO: Updating environment tmp-dev's configuration settings.
2018-07-11 21:06:50    INFO: Successfully deployed new configuration to environment.
2018-07-11 21:06:51    INFO: Environment update completed successfully.
```

以下命令将设置多个环境属性。它将添加名为 foo 的环境属性并将其值设置为 bar，更改 JDBC_CONNECTION_STRING 属性的值，并删除 PARAM4 和 PARAM5 属性。

```
$ eb setenv foo=bar JDBC_CONNECTION_STRING=hello PARAM4= PARAM5=
```

eb ssh

描述

> **Note**
>
> 此命令不适用于运行 Windows Server 实例的环境。

在环境中使用安全外壳（SSH）连接到 Linux Amazon EC2 实例。如果一个环境具有多个正在运行的实例，EB CLI 将提示您指定要连接到的实例。为了使用此命令，SSH 必须已安装到您的本地计算机上，并且必须可在命令行中使用。私有密钥文件必须位于用户目录中名为 `.ssh` 的文件夹下，且环境中的 EC2 实例必须拥有公有 IP 地址。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令还会连接到自定义环境中的实例。

> **SSH 密钥**
>
> 如果以前没有配置过 SSH，可以在运行 eb init 时使用 EB CLI 创建一个密钥。如果您已运行 eb init，请使用 `--interactive` 选项再次运行，并且在系统提示设置 SSH 时选择 Yes（是）和 Create New Keypair（创建新密钥对）。在此过程中创建的密钥将由 EB CLI 存储在适当的文件夹中。

对于从 0.0.0.0/0（所有 IP 地址）传入的流量，如果尚未实施端口 22 的任何规则，则该命令将临时打开您的环境的安全组中的端口 22。如果您已配置了环境的安全组，以便针对限定的 CIDR 范围打开端口 22 来提高安全性，EB CLI 将采用该设置并且放弃对安全组的任何更改。要覆盖此行为并且强制 EB CLI 对所有传入流量打开端口 22，请使用 `--force` 选项。

有关配置环境的安全组的信息，请参阅安全组。

语法

eb ssh

eb ssh ***environment-name***

选项

名称	描述

`-i`
或
`--instance`
	指定您连接到的实例的实例 ID。建议您使用此选项。

`-n`
或
`--number`
	指定要通过数字连接的实例。

`-o`
或
`--keep_open`
	在 SSH 会话结束后，将安全组上的端口 22 保留打开状态。

`--command`	在指定的实例上执行 SSH 命令，而不是启动 SSH 会话。
`--custom`	指定要使用的 SSH 命令，而不是 'ssh -i keyfile'。请勿包括远程用户和主机名。
`--setup`	更改分配到环境的实例的密钥对（需要替换实例）。
`--force`	在环境的安全组中针对来自 0.0.0.0/0 的传入流量打开端口 22，即使已为 SSH 配置安全组也是如此。 如果环境的安全组配置为针对限定的 CIDR 范围打开端口 22，而该范围不包括您尝试发出连接的 IP 地址，请使用此选项。
`--timeout` *`minutes`*	命令超时之前的设定分钟数。 只能与 `--setup` 参数一起使用。
常用选项	

输出

如果成功，命令将打开与实例的 SSH 连接。

示例

以下示例将您连接到指定环境。

```
$ eb ssh
Select an instance to ssh into
1) i-96133799
2) i-5931e053
(default is 1): 1
INFO: Attempting to open port 22.
INFO: SSH port 22 open.
The authenticity of host '54.191.45.125 (54.191.45.125)' can't be established.
RSA key fingerprint is ee:69:62:df:90:f7:63:af:52:7c:80:60:1b:3b:51:a9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.191.45.125' (RSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-8-185 ~]$ ls
[ec2-user@ip-172-31-8-185 ~]$ exit
logout
Connection to 54.191.45.125 closed.
INFO: Closed port 22 on ec2 instance security group
```

eb status

描述

提供有关环境状态的信息。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令还会提供关于生成器环境的信息。

语法

eb status

eb status ***environment-name***

选项

名称	描述

`-v`
或
`--verbose`
	提供有关单个实例的更多信息，例如其在 Elastic Load Balancing 负载均衡器上的状态。

常用选项	

输出

如果成功，命令将返回有关环境的以下信息：

- 环境名称
- 应用程序名称
- 部署的应用程序版本
- 环境 ID
- 平台
- 环境层
- 别名记录
- 环境的上次更新时间。
- 状态
- 运行状况

如果使用详细模式，EB CLI 还将提供正在运行的 Amazon EC2 实例的数目。

示例

以下示例显示了环境 tmp-dev 的状态。

```
$ eb status
Environment details for: tmp-dev
  Application name: tmp
  Region: us-west-2
  Deployed Version: None
  Environment ID: e-2cpfjbra9a
  Platform: 64bit Amazon Linux 2014.09 v1.0.9 running PHP 5.5
  Tier: WebServer-Standard-1.0
  CNAME: tmp-dev.elasticbeanstalk.com
  Updated: 2014-10-29 21:37:19.050000+00:00
  Status: Launching
  Health: Grey
```

eb swap

描述

将该环境的别名记录与另一个环境的别名记录交换（例如，为了在更新应用程序版本时避免停机）。

> **Note**
>
> 如果您拥有两个以上的环境，系统将提示您从环境列表中选择当前正在使用您所需的别名记录的环境的名称。要禁止此操作，您可以通过在运行命令时包括 -n 选项来指定要使用的环境的名称。

语法

eb swap

eb swap ***environment-name***

> **Note**
>
> *environment-name* 是您希望其具有其他别名记录的环境。如果您在运行 eb swap 时没有指定 *environment-name* 作为命令行参数，EB CLI 将更新默认环境的别名记录。

选项

名称	描述
-n 或 --destination_name	指定要与其交换别名记录的环境的名称。如果您运行不带此选项的 eb swap，EB CLI 将提示您从环境列表中选择。
常用选项	

输出

如果成功，则该命令返回 swap 操作的状态。

示例

以下示例将环境 tmp-dev 与 live-env 交换。

```
$ eb swap
Select an environment to swap with.
1) staging-dev
2) live-env
(default is 1): 2
2018-07-11 21:05:25    INFO: swapEnvironmentCNAMEs is starting.
2018-07-11 21:05:26    INFO: Swapping CNAMEs for environments 'tmp-dev' and 'live-env'.
2018-07-11 21:05:30    INFO: 'tmp-dev.elasticbeanstalk.com' now points to 'awseb-e-j-
AWSEBLoa-M7U21VXNLWHN-487871449.us-west-2.elb.amazonaws.com'.
2018-07-11 21:05:30    INFO: Completed swapping CNAMEs for environments 'tmp-dev' and
 'live-env'.
```

以下示例将环境 tmp-dev 与环境 live-env 交换，但不提示您输入或选择任何设置的值。

```
$ eb swap tmp-dev --destination_name live-env
2018-07-11 21:18:12    INFO: swapEnvironmentCNAMEs is starting.
2018-07-11 21:18:13    INFO: Swapping CNAMEs for environments 'tmp-dev' and 'live-env'.
2018-07-11 21:18:17    INFO: 'tmp-dev.elasticbeanstalk.com' now points to 'awseb-e-j-
AWSEBLoa-M7U21VXNLWHN-487871449.us-west-2.elb.amazonaws.com'.
2018-07-11 21:18:17    INFO: Completed swapping CNAMEs for environments 'tmp-dev' and
 'live-env'.
```

eb tags

描述

添加、删除、更新和列出 Elastic Beanstalk 资源的标签。

有关在 Elastic Beanstalk 中标记资源的详细信息，请参阅标记 Elastic Beanstalk 应用程序资源。

语法

eb tags [***environment-name***] [--resource ***ARN***] -l | --list

eb tags [***environment-name***] [--resource ***ARN***] -a | --add ***key1***=***value1***[,***key2***=***value2*** ...]

eb tags [***environment-name***] [--resource ***ARN***] -u | --update ***key1***=***value1***[,***key2***=***value2*** ...]

eb tags [***environment-name***] [--resource ***ARN***] -d | --delete ***key1***[,***key2*** ...]

您可以将 `--add`、`--update` 和 `--delete` 子命令选项组合在单个命令中。至少需要其中一个选项。这三个子命令选项中的任何一个都不能与 `--list` 结合使用。

不指定任何其他参数时，所有上述命令都将列出或修改当前目录的应用程序中默认环境的标签。指定 *environment-name* 参数时，命令将列出或修改此环境的标签。使用 `--resource` 选项时，命令将列出或修改任何 Elastic Beanstalk 资源的标签 - 应用程序、环境、应用程序版本、保存的配置或自定义平台版本。通过 Amazon Resource Name（ARN）指定资源。

选项

这些选项都不是必需的。如果运行不带任何选项的 eb create，则系统会提示您为每个设置输入或选择值。

名称	描述

`-l`
或
`--list`
	列出当前应用于资源的所有标签。

`-a` *key1*=*value1*[,*key2*=*value2*	将新标签应用到资源。将标签指定为逗号分隔的 `key=value` 对的列表。您无法指定现有标签的密钥。

名称	描述
或 `--add key1=value1[,key2=val`	有效值：请参阅为资源添加标签。

`-u key1=value1[,key2=value2`
 	
或
 	
`--updat e key1=value1[,key2=value2 .`
	更新现有资源标签的值。将标签指定为逗号分隔的 key=value 对的列表。您必须指定现有标签的密钥。
	
	有效值：请参阅为资源添加标签。

`-d key1[,key2 ...]`
 	
或
 	
`--delete key1[,key2 ...]`
	删除现有资源标签。将标签指定为逗号分隔的密钥列表。您必须指定现有标签的密钥。
	
	有效值：请参阅为资源添加标签。

`-r region`
 	
或
 	
`--region region`
	您的资源所在的AWS 区域区域。
	
	默认值：配置的默认区域。
	
	有关可以为此选项指定的值列表，请参阅 AWS 一般参考中的 AWS Elastic Beanstalk 端点和配额。

`--resource ARN`	要使用命令修改或列出其标签的资源的 ARN。如果未指定，则命令引用当前目录的应用程序中的（默认或指定）环境。 有效值：请参阅特定于您感兴趣的资源的为资源添加标签子主题。这些主题介绍资源 ARN 的构造方式，并说明如何为您的应用程序或账户获取此资源存在的 ARN 的列表。

输出

`--list` 子命令选项可显示资源标签的列表。输出既显示 Elastic Beanstalk 在默认情况下应用的标签，又显示您的自定义标签。

```
$ eb tags --list
Showing tags for environment 'MyApp-env':
```

```
Key                                 Value

Name                                MyApp-env
elasticbeanstalk:environment-id     e-63cmxwjaut
elasticbeanstalk:environment-name   MyApp-env
mytag                               tagvalue
tag2                                2nd value
```

--add、--update 和 --delete 子命令选项成功时没有任何输出。您可以通过添加 --verbose 选项来以查看命令活动的详细输出。

```
$ eb tags --verbose --update "mytag=tag value"
Updated Tags:

Key                                 Value

mytag                               tag value
```

示例

以下命令成功地将键为 tag1、值为 value1 的标签添加到应用程序的默认环境中，同时删除标签 tag2。

```
$ eb tags --add tag1=value1 --delete tag2
```

以下命令成功地将标签添加到应用程序中的保存的配置。

```
$ eb tags --add tag1=value1 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-
id:configurationtemplate/my-app/my-template"
```

以下命令失败，因为它尝试更新不存在的标签。

```
$ eb tags --update tag3=newval
ERROR: Tags with the following keys can't be updated because they don't exist:

  tag3
```

以下命令失败，因为它尝试更新和删除同一密钥。

```
$ eb tags --update mytag=newval --delete mytag
ERROR: A tag with the key 'mytag' is specified for both '--delete' and '--update'. Each
 tag can be either deleted or updated in a single operation.
```

eb terminate

描述

终止正在运行的环境，以便不会因未使用的 AWS 资源产生费用。

使用 `--all` 选项，删除当前目标已初始化为使用 eb init 的应用程序。该命令终止应用程序中的所有环境。它还会终止应用程序的应用程序版本和其保存的配置，然后删除应用程序。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令会终止正在运行的自定义环境。

> **Note**
>
> 稍后，您始终都可以使用相同的版本启动新的环境。

如果环境中有要保留的数据，请在终止环境之前将数据库删除策略设置为 `Retain`。这使数据库能够在 Elastic Beanstalk 之外运行。之后，任何 Elastic Beanstalk 环境都必须作为外部数据库连接到它。如果要在不保持数据库运行的情况下备份数据，请将删除策略设置为在终止环境之前拍摄数据库快照。有关更多信息，请参阅本指南的配置环境章节中的数据库生命周期。

> **Important**
>
> 如果您终止环境，则还必须删除您创建的任何 CNAME 映射，因为其他客户可能会重用可用的主机名。请务必删除指向已终止环境的 DNS 记录，以防出现 悬空 DNS 条目。悬空 DNS 条目可能会使指向您的域的互联网流量出现安全漏洞，此外还可能带来其他风险。
> 有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的 防止 Route 53 中悬挂委派记录。您还可以访问《AWS 安全博客》中的 针对 Amazon CloudFront 请求的增强域保护，了解有关悬空 DNS 条目的更多信息。

语法

eb terminate

eb terminate ***environment-name***

选项

名称	描述
`--all`	终止应用程序中的所有环境，终止应用程序的应用程序版本和其保存的配置，然后删除应用程序。
`--force`	终止环境，不提示确认。
`--ignore-links`	终止环境，即使它有链接的依赖环境。请参阅编写环境。
`--timeout`	命令超时之前的分钟数。

输出

如果成功，则该命令返回 `terminate` 操作的状态。

示例

以下示例请求终止环境 tmp-dev。

```
$ eb terminate
The environment "tmp-dev" and all associated instances will be terminated.
To confirm, type the environment name: tmp-dev
2018-07-11 21:05:25    INFO: terminateEnvironment is starting.
2018-07-11 21:05:40    INFO: Deleted CloudWatch alarm named: awseb-e-2cpfjbra9a-stack-
AWSEBCloudwatchAlarmHigh-16V08YOF2KQ7U
2018-07-11 21:05:41    INFO: Deleted CloudWatch alarm named: awseb-e-2cpfjbra9a-stack-
AWSEBCloudwatchAlarmLow-6ZAWH9F20P7C
2018-07-11 21:06:42    INFO: Deleted Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:5d7d3e6b-
d59b-47c5-b102-3e11fe3047be:autoScalingGroupName/awseb-e-2cpfjbra9a-stack-
AWSEBAutoScalingGroup-7AXY7U13ZQ6E:policyName/awseb-e-2cpfjbra9a-stack-AWSEBAutoSca
lingScaleUpPolicy-1876U27JEC34J
2018-07-11 21:06:43    INFO: Deleted Auto Scaling group policy named:
 arn:aws:autoscaling:us-east-2:11122223333:scalingPolicy:29c6e7c7-7ac8-46fc-91f5-
cfabb65b985b:autoScalingGroupName/awseb-e-2cpfjbra9a-stack-
AWSEBAutoScalingGroup-7AXY7U13ZQ6E:policyName/awseb-e-2cpfjbra9a-stack-AWSEBAutoSca
lingScaleDownPolicy-SL4LHODMOMU
```

```
2018-07-11 21:06:48    INFO: Waiting for EC2 instances to terminate. This may take a
 few minutes.
2018-07-11 21:08:55    INFO: Deleted Auto Scaling group named: awseb-e-2cpfjbra9a-
stack-AWSEBAutoScalingGroup-7AXY7U13ZQ6E
2018-07-11 21:09:10    INFO: Deleted security group named: awseb-e-2cpfjbra9a-stack-
AWSEBSecurityGroup-XT4YYGFL7I99
2018-07-11 21:09:40    INFO: Deleted load balancer named: awseb-e-2-AWSEBLoa-
AK6RRYFQVV3S
2018-07-11 21:09:42    INFO: Deleting SNS topic for environment tmp-dev.
2018-07-11 21:09:52    INFO: terminateEnvironment completed successfully.
```

eb upgrade

描述

将环境的平台升级到当前正在运行环境的平台的最新版本。

如果根目录包含一个指定自定义平台的 `platform.yaml` 文件，则该命令会将环境升级到当前正在运行环境的自定义平台的最新版本。

语法

eb upgrade

eb upgrade ***environment-name***

选项

名称	描述
`--force`	在开始升级过程之前无需您确认环境名称即可升级。
`--noroll`	在不使用滚动更新的情况下更新所有实例可在升级期间保留一些使用中的实例。
常用选项	

输出

该命令将显示更改概述并提示您通过键入环境名称来确认升级。如果成功，环境将更新并随最新版本的平台一起启动。

示例

以下示例将指定环境的当前平台版本升级到最新的可用平台版本。

```
$ eb upgrade
Current platform: 64bit Amazon Linux 2014.09 v1.0.9 running Python 2.7
Latest platform:  64bit Amazon Linux 2014.09 v1.2.0 running Python 2.7

WARNING: This operation replaces your instances with minimal or zero downtime. You may
 cancel the upgrade after it has started by typing "eb abort".
You can also change your platform version by typing "eb clone" and then "eb swap".

To continue, type the environment name:
```

eb use

描述

将指定环境设置为默认环境。

使用 Git 时，eb use 可为当前分支设置默认环境。请在每个您想部署到 Elastic Beanstalk 的分支中运行一次此命令。

语法

eb use ***environment-name***

选项

名称	描述
`--source codecommit/` *repository-name*`/`*branch-name*	CodeCommit 存储库和分支。请参阅通过 AWS CodeCommit 使用 EB CLI。
`-r` *region* `--region` *region*	更改您创建环境的区域。
常用选项	

常用选项

以下选项可用于所有 EB CLI 命令。

名称	描述
`--debug`	打印用于调试的信息。
`-h, --help`	显示帮助消息。 类型：字符串 默认值：无
`--no-verify-ssl`	跳过 SSL 证书验证。如果您在将 CLI 与代理结合使用时遇到问题，请使用此选项。
`--profile`	使用 AWS 凭证文件中的特定配置文件。
`--quiet`	隐藏来自命令的所有输出。
`--region`	使用指定区域。
`-v, --verbose`	显示详细信息。

EB CLI 2.6（已停用）

该版本的 EB CLI 及其文档已替换成版本 3 (在本部分中，EB CLI 3 表示 EB CLI 的版本 3 及更高版本)。有关新版本的信息，请参阅使用 Elastic Beanstalk 命令行界面 (EB CLI)。

您应该迁移到最新版本的 EB CLI 3。它可以管理您使用 EB CLI 2.6 或早期版本的 EB CLI 启动的环境。

与 EB CLI 版本 3 的区别

EB 是适用于 Elastic Beanstalk 的命令行界面 (CLI) 工具，您可以使用该工具更轻松地快速部署应用程序。Elastic Beanstalk 在 EB CLI 3 中引入了最新版本的 EB。EB CLI 将自动从使用 EB 创建的环境中检索设置 (如果该环境正在运行)。请注意，EB CLI 3 不像早期版本那样在本地存储选项设置。

EB CLI 引入了命令 eb create、eb deploy、eb open、eb console、eb scale、eb setenv、eb config、eb terminate、eb clone、eb list、eb use、eb printenv 和 eb ssh。在 EB CLI 3.1 或更高版本中，您还可以使用 eb swap 命令。仅在 EB CLI 3.2 中，您可使用 eb abort、eb platform 和 eb upgrade 命令。除了这些新命令之外，EB CLI 3 命令在某些情况下也不同于 EB CLI 2.6 命令：

- eb init – 使用 eb init 在现有项目的目录中创建一个 `.elasticbeanstalk` 目录，并为该项目创建新的 Elastic Beanstalk 应用程序。与旧版本不同，EB CLI 3 和更高版本不提示您创建环境。
- eb start – EB CLI 3 不包含命令 eb start。应使用 eb create 创建环境。
- eb stop – EB CLI 3 不包含命令 eb stop。应使用 eb terminate 完全终止环境并进行清理。
- eb push 和 **`git aws.push`** – EB CLI 3 不包含命令 eb push 或 `git aws.push`。应使用 eb deploy 更新应用程序代码。
- eb update – EB CLI 3 不包含命令 eb update。应使用 eb config 更新环境。
- eb branch – EB CLI 3 不包含命令 eb branch。

有关使用 EB CLI 3 命令创建和管理应用程序的更多信息，请参阅EB CLI 命令参考。有关如何使用 EB CLI 3 部署示例应用程序的演练，请参阅使用 EB CLI 管理 Elastic Beanstalk 环境。

迁移到 EB CLI 3 和 CodeCommit

Elastic Beanstalk 不仅弃用了 EB CLI 2.6，还删除了 2.6 的一些功能。与 2.6 相比的最大变化是 EB CLI 不再本机支持增量代码更新 (eb push、`git aws.push`) 或分支 (eb branch)。本节介绍如何从 EB CLI 2.6 迁移到最新版本的 EB CLI 并使用 CodeCommit 作为您的代码存储库。

如果您还没有在 CodeCommit 中创建代码存储库，请创建一个，如迁移到 CodeCommit 中所述。

安装并配置 EB CLI 后，您有两个机会将应用程序与 CodeCommit 存储库（包括特定分支）关联起来。

- 执行 eb init 时，如在以下示例中，*myRepo* 是您的 CodeCommit 存储库的名称，*myBranch* 是 CodeCommit 中的分支。

```
eb init --source codecommit/myRepo/myBranch
```

- 执行 eb deploy 时，如在以下示例中，*myRepo* 是您的 CodeCommit 存储库的名称，*myBranch* 是 CodeCommit 中的分支。

```
eb deploy --source codecommit/myRepo/myBranch
```

有关更多信息，包括如何在不必重新上传整个项目的情况下将增量代码更新部署到 Elastic Beanstalk 环境，请参阅通过 AWS CodeCommit 使用 EB CLI。

Elastic Beanstalk API 命令行界面（已停用）

此工具（Elastic Beanstalk API 命令行界面 (API CLI)）已由 AWS CLI 替代，后者提供了适用于所有 AWS服务的 API 等效命令。请参阅 AWS Command Line Interface 用户指南以开始使用 AWS CLI。另请尝试 EB CLI 以获取简化的高级命令行体验。

转换 Elastic Beanstalk API CLI 脚本

转换旧的 EB API CLI 脚本，以使用 AWS CLI 或 Tools for Windows PowerShell 来获取对最新的 Elastic Beanstalk API 的访问权限。下表列出 Elastic Beanstalk 基于 API 的 CLI 命令及其在 AWS CLI 和 Tools for Windows PowerShell 中的等效命令。

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
`elastic-beanstalk-check-dns-availability`	`check-dns-availability`	`Get-EBDNSAvailability`
`elastic-beanstalk-create-application`	`create-application`	`New-EBApplication`
`elastic-beanstalk-create-application-version`	`create-application-version`	`New-EBApplicationVersion`
`elastic-beanstalk-create-configuration-template`	`create-configuration-template`	`New-EBConfigurationTemplate`
`elastic-beanstalk-create-environment`	`create-environment`	`New-EBEnvironment`

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
elastic-beanstalk-create-storage-location	create-storage-location	New-EBStorageLocation
elastic-beanstalk-delete-application	delete-application	Remove-EBApplication
elastic-beanstalk-delete-application-version	delete-application-version	Remove-EBApplicationVersion
elastic-beanstalk-delete-configuration-template	delete-configuration-template	Remove-EBConfigurationTemplate
elastic-beanstalk-delete-environment-configuration	delete-environment-configuration	Remove-EBEnvironmentConfiguration
elastic-beanstalk-describe-application-versions	describe-application-versions	Get-EBApplicationVersion
elastic-beanstalk-describe-applications	describe-applications	Get-EBApplication
elastic-beanstalk-describe-configuration-options	describe-configuration-options	Get-EBConfigurationOption

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
elastic-beanstalk-describe-configuration-settings	describe-configuration-settings	Get-EBConfigurationSetting
elastic-beanstalk-describe-environment-resources	describe-environment-resources	Get-EBEnvironmentResource
elastic-beanstalk-describe-environments	describe-environments	Get-EBEnvironment
elastic-beanstalk-describe-events	describe-events	Get-EBEvent
elastic-beanstalk-list-available-solution-stacks	list-available-solution-stacks	Get-EBAvailableSolutionStack
elastic-beanstalk-rebuild-environment	rebuild-environment	Start-EBEnvironmentRebuild
elastic-beanstalk-request-environment-info	request-environment-info	Request-EBEnvironmentInfo
elastic-beanstalk-restart-app-server	restart-app-server	Restart-EBAppServer

Elastic Beanstalk API CLI	AWS CLI	AWS Tools for Windows PowerShell
elastic-beanstalk-retrieve-environment-info	retrieve-environment-info	Get-EBEnvironmentInfo
elastic-beanstalk-swap-environment-cnames	swap-environment-cnames	Set-EBEnvironmentCNAME
elastic-beanstalk-terminate-environment	terminate-environment	Stop-EBEnvironment
elastic-beanstalk-update-application	update-application	Update-EBApplication
elastic-beanstalk-update-application-version	update-application-version	Update-EBApplicationVersion
elastic-beanstalk-update-configuration-template	update-configuration-template	Update-EBConfigurationTemplate
elastic-beanstalk-update-environment	update-environment	Update-EBEnvironment
elastic-beanstalk-validate-configuration-settings	validate-configuration-settings	Test-EBConfigurationSetting

AWS Elastic Beanstalk 安全性

AWS的云安全性的优先级最高。作为AWS客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是AWS和您的共同责任。责任共担模式将其描述为云的安全性 和云中的安全性。

云的安全性 – AWS负责保护运行在AWS云中提供的所有服务的基础设施，并向您提供可安全使用的服务。我们的安全责任是AWS中的最高优先级，作为 AWS合规性计划的一部分，我们安全性的有效性由第三方审计员定期进行测试和验证。查看AWS保证计划范围内的AWS服务，了解与 Elastic Beanstalk 相关的信息。

云中的安全性 – 您的责任由您所使用的AWS服务以及其他因素决定，包括您数据的敏感性、您组织的要求以及适用的法律法规。本文档旨在帮助您了解如何在使用 Elastic Beanstalk 时应用责任共担模式。

使用以下安全主题详细了解 Elastic Beanstalk 负责的安全任务，以及在使用 Elastic Beanstalk 来实现安全性和合规性目标时应考虑的安全配置。

主题

Elastic Beanstalk 中的数据保护

AWS责任共担模式适用于 AWS Elastic Beanstalk 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如 Name（名称）字段）。这包括使用控制台、API、AWS CLI 或 AWS 开发工具包处理 Elastic Beanstalk 或其他 AWS 服务的情况。您在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

主题

使用加密保护数据

Elastic Beanstalk 将各种对象存储在 Amazon Simple Storage Service (Amazon S3) 存储桶中，对于您在其中创建环境的每个 AWS 区域都将创建一个存储桶。有关详细信息，请参阅the section called “Amazon S3”。

您提供一些存储对象，并将它们发送到 Elastic Beanstalk，例如，应用程序版本和源包。Elastic Beanstalk 会生成其他对象，例如日志文件。除了 Elastic Beanstalk 存储的数据之外，您的应用程序还可以传输和/或存储数据作为其操作的一部分。

数据保护是指，在数据传输（传入和传出 Elastic Beanstalk 时）和处于静态（存储在 AWS 数据中心时）期间保护数据。

传输中加密

您可以通过两种方式在传输过程中实现数据保护：使用安全套接字层 (SSL) 加密连接，或使用客户端加密（对象在发送之前先进行加密）。这两种方法都可有效地保护您的应用程序数据。为了保护连接，请在您的应用程序、其开发人员和管理员以及最终用户发送或接收任何对象时使用 SSL 对其进行加密。有关加密往返于您的应用程序的 Web 流量的详细信息，请参阅the section called “HTTPS”。

客户端加密不是用于保护您上传的应用程序版本和源包中的源代码的有效方法。Elastic Beanstalk 需要访问这些对象，因此无法对其进行加密。因此，请确保保护开发或部署环境与 Elastic Beanstalk 之间的连接。

静态加密

要保护应用程序的静态数据，请了解应用程序使用的存储服务中的数据保护。例如，请参阅《Amazon RDS 用户指南》中的 Amazon RDS 中的数据保护、《Amazon Simple Storage Service 用户指南》中的 Amazon S3 中的数据保护或《Amazon Elastic File System 用户指南》中的在 EFS 中加密数据和元数据。

Elastic Beanstalk 不会为其创建的 Amazon S3 存储桶启用默认加密。这意味着，在默认情况下，对象以未加密形式存储在存储桶中（并且只有经授权可以读取此存储桶的用户可以访问）。如果您的应用程序需要静态加密，可以将您的账户的存储桶配置为默认加密。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 默认 S3 存储桶加密。

有关数据保护的更多信息，请参阅AWS安全性博客上的AWS责任共担模式和 GDPR 博客文章。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

互联网络流量隐私

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 在 Elastic Beanstalk 应用程序中的资源之间创建边界，并控制它们、本地网络和 Internet 之间的流量。有关详细信息，请参阅the section called “Amazon VPC”。

有关 Amazon VPC 安全性的更多信息，请参阅 Amazon VPC 用户指南 中的 安全性。

有关数据保护的更多信息，请参阅AWS安全性博客上的AWS责任共担模式和 GDPR 博客文章。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

适用于 Elastic Beanstalk 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一种 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 AWS Elastic Beanstalk 资源。IAM 是一个可以免费使用的 AWS 服务。

有关使用 IAM 的详细信息，请参阅将 Elastic Beanstalk 与 AWS Identity and Access Management 配合使用。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

AWS 的托管策略 AWS Elastic Beanstalk

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

Elastic Bean AWS stalk 更新了托管策略

查看有关 Elastic Beanstalk AWS 托管策略自 2021 年 3 月 1 日以来更新的详细信息。

要查看特定托管策略的 JSON 来源，请参阅AWS 托管策略参考指南。

更改	描述	日期
以下政策已更新：	这些政策已更新，允许 Elastic Beanstalk 在创建或 AWS CloudFormation 更新堆栈或更改集时添加或删除标签。	2024 年 4 月 30 日

更改	描述	日期
• AWSElasticBeanstalkInternalMaintenanceRolePolicy • AWSElasticBeanstalkMaintenance • AWSElasticBeanstalkManagedUpdatesInternalServiceRolePolicy • AWSElasticBeanstalkManagedUpdatesServiceRolePolicy • AWSElasticBeanstalkRoleCore	有关 AWSElasticBeanstalkManagedUpdatesServiceRolePolicy 的更多信息，请参阅 Elastic Beanstalk 的服务相关角色权限。 有关 AWSElasticBeanstalkRoleCore 的更多信息，请参阅 与其他服务集成的策略。	

更改	描述	日期
AWSElasticBeanstalkService—更新了现有政策	此策略已更新，允许 Elastic Beanstalk 在为 Elastic Load Balancing、自动扩缩组（ASG）和 Amazon ECS 创建资源时，向资源添加标签。 **Note** 此策略之前已被 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 取代。尽管此策略不再能附加到新的 IAM 用户、组或角色，但仍可附加到之前已经存在的用户、组或角色。 有关详细信息，请参阅托管服务角色策略。	2023 年 5 月 10 日
AWSElasticBeanstalkMulticontainerDocker—更新了现有政策	此策略已更新，以允许 Elastic Beanstalk 在创建 Amazon ECS 资源时向资源添加标签。 有关更多信息，请参阅 管理 Elastic Beanstalk 实例配置文件。	2023 年 3 月 23 日

更改	描述	日期
AWSElasticBeanstalkRoleECS—更新了现有政策	此策略已更新，以允许 Elastic Beanstalk 在创建 Amazon ECS 资源时向资源添加标签。 有关更多信息，请参阅 与其他服务集成的策略。	2023 年 3 月 23 日
AdministratorAccess-AWSElasticBeanstalk —更新了现有政策	此策略已更新，以允许 Elastic Beanstalk 在创建 Amazon ECS 资源时向资源添加标签。 有关更多信息，请参阅 管理 Elastic Beanstalk 用户策略。	2023 年 3 月 23 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy —更新了现有政策	此策略已更新，以允许 Elastic Beanstalk 在创建 Amazon ECS 资源时向其添加标签。 有关更多信息，请参阅 Elastic Beanstalk 的服务相关角色权限。	2023 年 3 月 23 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy—更新了现有政策	此策略已更新，以允许 Elastic Beanstalk 在创建 Amazon ECS 资源时向其添加标签。 有关详细信息，请参阅托管服务角色策略。	2023 年 3 月 23 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy—更新了现有政策	此策略已更新，允许 Elastic Beanstalk 在创建自动扩缩组时向其添加标签。 有关更多信息，请参阅 托管更新服务相关角色。	2023 年 1 月 27 日

更改	描述	日期
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy—更新了现有政策	此策略已更新，允许 Elastic Beanstalk 在创建自动扩缩组（ASG）时添加标签。 有关详细信息，请参阅托管服务角色策略。	2023 年 1 月 23 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy—更新了现有政策	此策略已更新，允许 Elastic Beanstalk 在创建弹性负载均衡器 (ELB) 时添加标签。 有关详细信息，请参阅托管服务角色策略。	2022 年 12 月 21 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy—更新了现有政策	向此策略添加权限，以允许 Elastic Beanstalk 在托管更新期间执行以下操作： • 创建并删除启动模板和模板版本。 • 使用启动模板启动 Amazon EC2 实例。 • 如果存在 Amazon RDS，则检索可用数据库引擎列表以及有关预置 RDS 实例的信息。 有关更多信息，请参阅 托管更新服务相关角色。	2022 年 8 月 23 日

更改	描述	日期
AWSElasticBeanstalkReadOnlyAccess— 已弃用 GovCloud （美国） AWS 区域	此策略已被 `AWSElasticBeanstalkReadOnly` 取代。 该政策将在 GovCloud （美国）逐步取消 AWS 区域。 当此策略逐步淘汰后，它在 2021 年 6 月 17 日之后将无法再挂载到新的 IAM 用户、组或角色。 有关更多信息，请参阅用户策略。	2021 年 6 月 17 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy—更新了现有政策	此策略已更新，允许 Elastic Beanstalk 读取 EC2 可用区的属性。它使 Elastic Beanstalk 能够跨可用区更有效地验证您的实例类型选择。 有关详细信息，请参阅托管服务角色策略。	2021 年 6 月 16 日

更改	描述	日期
AWSElasticBeanstalkFullAccess— 已弃用 GovCloud （美国） AWS 区域	此策略已被 `AdministratorAccess-AWSElasticBeanstalk` 取代。 该政策将在 GovCloud （美国）逐步取消 AWS 区域。 当此策略逐步淘汰后，它在 2021 年 6 月 10 日之后将无法再挂载到新的 IAM 用户、组或角色。 有关更多信息，请参阅用户策略。	2021 年 6 月 10 日
以下托管策略已在中国所有国家中被弃用 AWS 区域： • AWSElasticBeanstalkFullAccess • AWSElasticBeanstalkReadOnlyAccess	`AWSElasticBeanstalkFullAccess` 策略已被 `AdministratorAccess-AWSElasticBeanstalk` 取代。 `AWSElasticBeanstalkReadOnlyAccess` 策略已被 `AWSElasticBeanstalkReadOnly` 取代。 这些政策已在中国所有国家逐步取 AWS 区域消。 2021 年 6 月 3 日后，这些策略将无法再挂载到新的 IAM 用户、组或角色。 有关更多信息，请参阅用户策略。	2021 年 6 月 3 日

更改	描述	日期
AWSElasticBeanstalkService— 已弃用	此策略已被 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 取代。 此策略已被逐步淘汰，它将无法再挂载到新的 IAM 用户、组或角色。 有关详细信息，请参阅托管服务角色策略。	2021 年 6 月 - 2022 年 1 月
除中国和 GovCloud （美国）外，以下托管策略已在所有 AWS 区域版本中弃用： • AWSElasticBeanstalkFullAccess • AWSElasticBeanstalkReadOnlyAccess	`AWSElasticBeanstalkFullAccess` 策略已被 `AdministratorAccess-AWSElasticBeanstalk` 取代。 `AWSElasticBeanstalkReadOnlyAccess` 策略已被 `AWSElasticBeanstalkReadOnly` 取代。 除中国和 GovCloud （美国）外，这些政策已在所有国家逐步取消。 AWS 区域 2021 年 4 月 16 日后，这些策略将无法再挂载到新的 IAM 用户、组或角色。 有关更多信息，请参阅用户策略。	2021 年 4 月 16 日

更改	描述	日期
以下托管策略已更新： • AdministratorAccess-AWSElasticBeanstalk • AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy	这两个策略现在都支持中国的 PassRole 权限 AWS 区域。 有关 `AdministratorAccess-AWSElasticBeanstalk` 的详细信息，请参阅用户策略。 有关 `AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy` 的详细信息，请参阅托管服务角色策略。	2021 年 3 月 9 日
AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy：新策略	Elastic Beanstalk 增加了一个新策略，用于取代 `AWSElasticBeanstalkService` 托管策略。 这项新托管策略通过应用一组限制性更强的权限来提高资源的安全性。 有关详细信息，请参阅托管服务角色策略。	2021 年 3 月 3 日
Elastic Beanstalk 开始跟踪变更	Elastic Beanstalk 开始跟踪托管策略的变更。 AWS	2021 年 3 月 1 日

Elastic Beanstalk 中的日志记录和监控

要维护 AWS Elastic Beanstalk 和您的AWS解决方案的可靠性、可用性和性能，实施监控非常重要。您应该从AWS解决方案的各个部分收集监控数据，以便您可以更轻松地调试多点故障（如果发生）。AWS提供了多种工具来监控您的 Elastic Beanstalk 资源并对潜在事件做出响应。

有关监控的更多信息，请参阅监控环境。

有关其他 Elastic Beanstalk 安全主题，请参阅 AWS Elastic Beanstalk 安全性。

增强型运行状况报告

增强型运行状况报告是一种功能，您可以在您的环境中启用该功能以允许 Elastic Beanstalk 收集有关环境中的资源的其他信息。Elastic Beanstalk 分析信息，以更好地了解总体环境运行状况并帮助识别可能导致您的应用程序变得不可用的问题。有关更多信息，请参阅 增强型运行状况报告和监控。

Amazon EC2 实例日志

Elastic Beanstalk 环境中的 Amazon EC2 实例会生成日志，您可以查看这些日志来对应用程序或配置文件进行故障排除。由 Web 服务器、应用程序服务器、Elastic Beanstalk 平台脚本和 AWS CloudFormation 创建的日志在本地存储在各个实例上。您可使用环境管理控制台或 EB CLI 轻松检索这些日志。您还可以将环境配置为将日志实时流式传输到 Amazon CloudWatch Logs。有关更多信息，请参阅查看您的 Elastic Beanstalk 环境中的 Amazon EC2 实例的日志。

环境通知

您可以配置 Elastic Beanstalk 环境，以使用 Amazon Simple Notification Service (Amazon SNS) 向您通知影响应用程序的重要事件。您可以在创建环境期间或在创建环境后指定一个电子邮件地址，以便在发生错误或环境的运行状况发生变化时从AWS接收电子邮件。有关更多信息，请参阅使用 Amazon SNS 发送 Elastic Beanstalk 环境通知。

Amazon CloudWatch 警报

使用 CloudWatch 警报，可以观看单个指标在指定时间段内的变化。如果指标超出给定阈值，将向 Amazon SNS 主题或 AWS Auto Scaling 策略发送通知。CloudWatch 警报不会仅仅因为它们处于特定状态而调用操作。相反，当状态更改并维持指定的时间段时，警报会调用操作。有关更多信息，请参阅将 Elastic Beanstalk 和 Amazon CloudWatch 结合使用。

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或AWS服务在 Elastic Beanstalk 中所执行操作的记录。使用 CloudTrail 收集的信息，您可以确定向 Elastic Beanstalk 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Elastic Beanstalk API 调用。

调试 AWS X-Ray

X-Ray 是一种AWS服务，用于收集有关应用程序服务的请求的数据，并使用它来构建服务地图，以便您发现应用程序问题和优化机会。您可以使用 AWS Elastic Beanstalk 控制台或配置文件在环境中的实例上运行 X-Ray 守护程序。有关更多信息，请参阅配置 AWS X-Ray 调试。

Elastic Beanstalk 的合规性验证

作为多个AWS合规性计划的一部分，第三方审计员将评估 AWS Elastic Beanstalk 的安全性和合规性。这些计划包括 SOC、PCI、FedRAMP、HIPAA 等。AWS在特定合规性计划范围内提供经常更新的AWS服务列表，包括合规性计划范围内的AWS服务。

第三方审计报告可供您使用 AWS Artifact 进行下载。有关更多信息，请参阅在 AWS Artifact 中下载报告。

有关AWS合规性计划的更多信息，请参阅 AWS合规性计划。

您在使用 Elastic Beanstalk 时的合规性责任由您数据的敏感性、您组织的合规性目标以及适用的法律法规决定。如果您对 Elastic Beanstalk 的使用需遵守 HIPAA、PCI 或 FedRAMP 等标准，AWS将提供以下有用资源：

- 安全性与合规性快速入门指南 – 部署指南讨论了架构注意事项，并提供了在AWS上部署侧重安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书描述公司如何使用AWS创建符合 HIPAA 标准的应用程序。
- AWS合规性资源 – 这是合规性业务手册和指南集合，可能适用于您所在的行业和场所。
- AWS Config – 一项服务，可评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – AWS中安全状况的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

Elastic Beanstalk 中的弹性

AWS全球基础设施围绕AWS区域和可用区构建。

AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。

利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关AWS区域和可用区的更多信息，请参阅AWS全球基础设施。

AWS Elastic Beanstalk 代表您管理和自动执行AWS全球基础架构的使用。使用 Elastic Beanstalk 时，您将从AWS提供的可用性和容错机制中受益。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

Elastic Beanstalk 中的基础设施安全性

作为一项托管服务，AWS Elastic Beanstalk 由亚马逊云科技：安全流程概述白皮书中所述的AWS全球网络安全程序提供保护。

您可以使用AWS发布的 API 调用来通过网络访问 Elastic Beanstalk。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代平台（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

Elastic Beanstalk 中的配置和漏洞分析

AWS和我们的客户共担实现高水平的软件组件安全性和合规性的责任。AWS Elastic Beanstalk 通过提供托管更新功能来帮助您履行共同责任模型中您的责任。此功能会自动为 Elastic Beanstalk 支持的平台版本应用补丁更新和次要版本更新。

有关更多信息，请参阅 Elastic Beanstalk 平台维护的责任共担模型。

有关其他 Elastic Beanstalk 安全主题，请参阅AWS Elastic Beanstalk 安全性。

Elastic Beanstalk 的安全最佳实践

AWS Elastic Beanstalk 提供了您在开发和实施自己的安全策略时需要考虑的多项安全功能。以下最佳实践是一般准则，并不代表完整的安全解决方案。由于这些最佳实践可能不适合您的环境或不满足您的环境要求，因此将其视为有用的考虑因素，而不是惯例。

有关其他 Elastic Beanstalk 安全主题，请参阅 AWS Elastic Beanstalk 安全性。

预防性安全最佳实践

预防性安全控制措施试图在事件发生之前进行预防。

实施最低权限访问

Elastic Beanstalk 为实例配置文件、服务角色和 IAM 用户提供 AWS Identity and Access Management (IAM) 托管策略。这些托管策略指定了正确运行环境和应用程序可能所需的所有权限。

您的应用程序可能不需要我们托管策略中的全部权限。您可以自定义它们，并仅授予环境的实例、Elastic Beanstalk 服务和用户执行其任务所需的权限。这与用户策略特别相关，其中不同的用户角色可能具有不同的权限需求。实施最低权限访问对于减小安全风险以及可能由错误或恶意意图造成的影响至关重要。

定期更新您的平台

Elastic Beanstalk 定期发布新平台版本更新其所有平台。新的平台版本提供了操作系统、运行时、应用程序服务器和 Web 服务器更新以及 Elastic Beanstalk 组件更新。其中很多平台更新包括重要的安全修复。确保您的 Elastic Beanstalk 环境在受支持的平台版本（通常是平台的最新版本）上运行。有关详细信息，请参阅 更新 Elastic Beanstalk 环境的平台版本。

保持环境平台最新的最简单方法是将环境配置为使用托管平台更新。

在环境实例上强制执行 IMDSv2

您的 Elastic Beanstalk 环境中的 Amazon Elastic Compute Cloud (Amazon EC2) 实例使用实例元数据服务 (IMDS)（实例上的一个组件）来安全地访问实例元数据。IMDS 支持两种访问数据的方法：IMDSv1 和 IMDSv2。IMDSv2 使用面向会话的请求，并防范了多种类型的漏洞，这些漏洞可用于尝试访问 IMDS。有关 IMDSv2 优势的详细信息，请参阅为 EC2 实例元数据服务增加深度防御的增强功能。

IMDSv2 更加安全，因此最好在您的实例上强制使用 IMDSv2。要强制使用 IMDSv2，请确保应用程序的所有组件均支持 IMDSv2，然后禁用 IMDSv1。有关更多信息，请参阅 the section called “IMDS”。

检测性安全最佳实践

侦探性安全控件会在发生安全违规后识别它们。它们可以帮助您发现潜在安全威胁或事件。

实施监控

监控是保持您的 Elastic Beanstalk 解决方案的可靠性、安全性、可用性和性能的重要部分。AWS为您提供了一些可以用来监控AWS服务的工具。

以下是要监视的项目的一些示例：

- 适用于 Elastic Beanstalk 的 Amazon CloudWatch 指标 – 为关键 Elastic Beanstalk 指标和应用程序的自定义指标设置警报。有关详细信息，请参阅 将 Elastic Beanstalk 和 Amazon CloudWatch 结合使用。
- AWS CloudTrail 条目 – 跟踪可能影响可用性的操作，例如 `UpdateEnvironment` 或 `TerminateEnvironment`。有关详细信息，请参阅 使用 AWS CloudTrail 记录 Elastic Beanstalk API 调用。

启用 AWS Config

AWS Config 可以提供关于您的账户中的AWS资源配置的详细信息。您可以查看资源的关联方式、获取配置更改的历史记录并了解关系和配置如何随时间的推移而变化。

您可以使用 AWS Config 定义评估资源配置是否符合数据的规则。AWS Config 规则代表 Elastic Beanstalk 资源的理想配置设置。如果某个资源违反了某规则并且被标记为“不合规”，则 AWS Config 可能提醒您使用 Amazon Simple Notification Service (Amazon SNS) 主题。有关详细信息，请参阅 使用 查找和跟踪 Elastic Beanstalk 资源AWS Config。

问题排查

本章提供关于对 Elastic Beanstalk 环境进行故障排除的指导。其中提供以下信息。

- AWS Systems Manager 工具简介，以及预定义 Elastic Beanstalk 运行手册的运行步骤，该运行手册将提供故障排除步骤和建议。
- 关于在环境状态降级时可采取的措施及可查看的资源的一般指导。
- 按主题类别提供的更具体故障排除提示。

如果您的环境运行状况变为红色，建议您首先使用包含预定义运行手册的 AWS Systems Manager 工具，对 Elastic Beanstalk 进行故障排除。有关更多信息，请参阅本章后续部分中的 使用 Systems Manager 工具。

主题

使用 AWS Systems Manager Elastic Beanstalk 运行手册

您可以使用 Systems Manager 对 Elastic Beanstalk 环境进行故障排除。为帮助您快速入门，Systems Manager 提供预定义的 Elastic Beanstalk 自动化运行手册。自动化运行手册是一种 Systems Manager 文档，用于定义要对环境的实例及其他 AWS 资源执行的操作。

`AWSSupport-TroubleshootElasticBeanstalk` 文档是一本自动化运行手册，旨在帮助识别一系列可能导致 Elastic Beanstalk 环境降级的常见问题。为此，它会检查环境组件，包括以下组件：EC2 实例、VPC、AWS CloudFormation 堆栈、负载均衡器、自动扩缩组以及与安全组规则、路由表和 ACL 关联的网络配置。

它还会提供将捆绑日志文件从环境上传到 AWS Support 的选项。

有关更多信息，请参阅 AWS Systems Manager Automation 运行手册参考中的 `AWSSupport-TroubleshootElasticBeanstalk`。

通过 Systems Manager 运行 **AWSSupport-TroubleshootElasticBeanstalk** 运行手册

> **Note**
>
> 在 Elastic Beanstalk 环境所在的同一 AWS 区域 中运行此过程。

1. 打开 AWS Systems Manager 控制台。
2. 从导航窗格中的变更管理下，选择自动化。
3. 选择执行自动化。
4. 在 Amazon 所有选项卡的自动化文档搜索框中，输入 `AWSSupport-TroubleshootElasticBeanstalk`。
5. 选择 AWSSupport-TroubleshootElasticBeanstalk 选项卡，然后选择下一步。
6. 选择执行。
7. 在输入参数部分中：
   a. 从 AutomationAssumeRole 下拉列表中，选择允许 Systems Manager 代表您执行操作的角色的 ARN。
   b. 在 ApplicationName中，输入 Elastic Beanstalk 应用程序的名称。
   c. 在环境名称中，输入 Elastic Beanstalk 环境。
   d. （可选）对于 S3UploaderLink，如果 AWS Support 工程师向您提供了用于日志收集的 S3 链接，请输入链接。
8. 选择执行。

   如果任一步骤失败，请选择该失败步骤的步骤 ID 列下的链接。这将显示该步骤的执行详细信息页面。VerificationErrorMessage 部分则会显示需要注意的步骤摘要。例如，`IAMPermissionCheck` 可能会显示警告消息。在这种情况下，您可以检查 AutomationAssumeRole 下拉列表中选择的角色是否具有必要的权限。

成功完成所有步骤后，输出会提供故障排除步骤和建议，以将环境恢复到正常运行状态。

一般指南

错误消息可显示在控制台的“事件”页面、日志中或“运行状况”页面上。您也可以采取措施从最近更改所导致的降级环境中恢复。如果您的环境运行状况变为红色，请尝试以下操作：

- 检查最近的环境事件。来自 Elastic Beanstalk 的有关部署、加载和配置问题的消息通常出现在此处。
- 检查最近的环境更改历史记录。更改历史记录列出了对环境进行的所有配置更改，并包括其他信息，例如哪些 IAM 用户进行了更改以及设置了哪些配置参数。
- 拉取日志以查看最近的日志文件条目。Web 服务器日志包含有关传入请求和错误的信息。
- 连接到实例并检查系统资源。
- 回滚到应用程序的前一个正常工作的版本。
- 撤消最近的配置更改或还原已保存的配置。
- 部署新环境。如果环境看起来运行状况良好，则执行别名记录交换以将流量路由到新环境并继续调试旧环境。

类别

本主题将按类别提供更具体的故障排除提示。

主题

- 连接
- 环境创建和实例启动
- 部署
- 健康
- 配置
- Docker 容器故障排除
- 常见问题

连接

问题：Elastic Beanstalk 控制台中创建的服务器不会显示在 Toolkit for Eclipse 中

您可以按照将现有的环境导入 Eclipse中的说明手动导入服务器。

问题：无法从 Elastic Beanstalk 连接到 Amazon RDS。

要将解耦的 Amazon RDS 连接到 Elastic Beanstalk 应用程序，请执行以下操作：

- 确保 RDS 与 Elastic Beanstalk 应用程序位于同一区域。
- 确保您的实例的 RDS 安全组拥有相关授权，从而使您正在使用的 Amazon EC2 安全组可用于您的 Elastic Beanstalk 环境。有关如何使用 AWS 管理控制台查找 EC2 安全组名称的说明，请参阅安全组。有关配置 EC2 安全组的详细信息，请转到 Amazon Relational Database Service 用户指南 中的使用数据库安全组的“向 Amazon EC2 安全组授予网络访问权限”部分。
- 对于 Java，确保您的 WEB–INF/lib 中含有 MySQL JAR 文件。有关更多信息，请参阅 向 Java 应用程序环境中添加 Amazon RDS 数据库实例。

环境创建和实例启动

事件：启动环境失败

此事件在 Elastic Beanstalk 尝试启动环境并在过程中遭遇失败时发生。事件页上过去的事件会提醒您此问题的根本原因。

事件：环境创建完成，但出现命令超时。请尝试增加超时时间。

如果您使用的配置文件要对实例运行命令、下载大型文件或安装程序包，则部署应用程序的时间可能很长。增加命令超时，让应用程序在部署期间有更多时间开始运行。

事件：无法创建以下资源：[AWSEBInstanceLaunchWaitCondition]

此消息指示您环境的 Amazon EC2 实例未与成功启动的 Elastic Beanstalk 通信。如果实例没有 Internet 连接就会出现这种情况。如果您配置了环境以在私有 VPC 子网中启动实例，请确保子网具有 NAT 以允许实例连接到 Elastic Beanstalk。

事件：此区域中需要服务角色。请将服务角色选项添加到环境。

Elastic Beanstalk 使用服务角色来监视环境中的资源并支持托管平台更新。参阅 管理 Elastic Beanstalk 服务角色 了解更多信息。

部署

问题：应用程序在部署期间变得不可用

由于 Elastic Beanstalk 使用直接升级过程，因此可能会有几秒钟的停机时间。使用滚动部署可尽量减少部署对您的生产环境的影响。

事件：无法创建 AWS Elastic Beanstalk 应用程序版本

应用程序源包可能太大，或者您已达到了应用程序版本配额。

事件：环境更新完成，但出现命令超时。请尝试增加超时时间。

如果您使用的配置文件要对实例运行命令、下载大型文件或安装程序包，则部署应用程序的时间可能很长。增加命令超时，让应用程序在部署期间有更多时间开始运行。

健康

事件：CPU 使用率超过 95.00%

尝试运行更多实例或选择不同的实例类型。

事件：Elastic Load Balancer awseb-*myapp* 没有正常运行的实例

如果您的应用程序确实在工作，请确保正确配置了应用程序的运行状况检查 URL。如果不是这种情况，请查看运行状况屏幕和环境日志以了解更多信息。

事件：找不到 Elastic Load Balancer awseb-*myapp*

您的环境负载均衡器可能已经在外部删除。仅使用配置选项以及 Elastic Beanstalk 提供的可扩展性对环境资源进行更改。重建您的环境或者启动新环境。

事件：EC2 实例启动失败。等待新的 EC2 实例启动……

您环境的实例类型可能具有低可用性，或者您已达到了账户的实例配额。检查服务运行状况控制面板，以确保 Elastic Compute Cloud (Amazon EC2) 服务为绿色，或者请求提高配额。

配置

事件：您无法使用 Elastic Load Balancing Target（Elastic Load Balancing 目标）选项和 Application Healthcheck URL（应用程序运行状况检查 URL）选项的值来配置 Elastic Beanstalk 环境

`Target` 命名空间中的 `aws:elb:healthcheck` 选项已弃用。从环境中删除 `Target` 选项命名空间，然后重试更新。

事件：ELB 无法连接到一可用区的多个子网。

如果您尝试在同一可用区的子网之间移动负载均衡器，则会看到此消息。更改负载均衡器上的子网需要将其移出原始可用区，然后将所需子网移回原始可用区。在此过程中，您的所有实例都将在可用区之间迁移，从而导致明显的停机时间。相反，请考虑创建新环境并执行别名记录交换。

Docker 容器故障排除

事件：无法拉取最新 Docker 映像：资源库名称 () 无效，只允许 [a-z0-9-_.]。有关详细信息，请跟踪日志。

使用 JSON 验证程序检查 `dockerrun.aws.json` 文件的语法。另请针对 Docker 配置 中说明的要求验证 dockerfile 内容

事件：在 Dockerfile 中未找到 EXPOSE 指令，中止部署

`Dockerfile` 或 `dockerrun.aws.json` 文件未声明容器端口。使用 `EXPOSE` 指令 (`Dockerfile`) 或 `Ports` 块 (`dockerrun.aws.json` 文件) 为传入流量公开端口。

事件：无法从#####下载身份验证凭证###

`dockerrun.aws.json` 为 `.dockercfg` 文件提供了无效的 EC2 密钥对和/或 S3 存储桶。或者，实例配置文件没有针对 S3 存储桶的 GetObject 授权。验证 `.dockercfg` 文件是否包含有效的 S3 存储桶和 EC2 密钥对。在实例配置文件中授予 IAM 角色执行 `s3:GetObject` 操作的权限。有关详细信息，请转到 管理 Elastic Beanstalk 实例配置文件

事件： 活动执行失败，原因是出现警告：身份验证配置文件无效

您的身份验证文件 (`config.json`) 格式不正确。请参阅 使用私有存储库中的映像。

常见问题

问题：如何将我的应用程序 URL 从 myapp.us-west-2.elasticbeanstalk.com 更改为 www.myapp.com？

在 DNS 服务器中注册一个别名记录，如 **`www.mydomain.com CNAME mydomain.elasticbeanstalk.com`**。

问题：如何为我的 Elastic Beanstalk 应用程序指定一个特定可用区。

您可以使用 API、CLI、Eclipse 插件或 Visual Studio 插件来选取特定可用区。有关使用 Elastic Beanstalk 控制台指定可用区的说明，请参阅Elastic Beanstalk 环境的 Auto Scaling 组。

问题：如何更改我的环境的实例类型？

要更改环境的实例类型，请转到环境配置页面，然后在实例配置类别中选择编辑。然后，选择一种新的实例类型并选择 Apply（应用）以更新环境。之后，Elastic Beanstalk 终止所有正在运行的实例，并将之替换为新实例。

问题：如何确定是否有人对环境进行了配置更改？

要查看此信息，请在 Elastic Beanstalk 控制台的导航窗格中选择 Change history（更改历史记录）以显示所有环境的配置更改列表。此列表包括更改的日期和时间、更改的配置参数及值以及进行更改的 IAM 用户。有关更多信息，请参阅更改历史记录。

问题：能否防止在实例终止时删除 Amazon EBS 卷？

您环境中的实例使用 Amazon EBS 进行存储；不过，当 Auto Scaling 终止一个实例时，根卷会被删除。我们不建议您在实例上存储状态或其他数据。如果需要，您可以使用 AWS CLI 阻止卷被删除：`$ aws ec2 modify-instance-attribute -b '/dev/sdc=<vol-id>:false`；具体说明请参阅 AWS CLI 参考。

问题：如何删除 Elastic Beanstalk 应用程序中的个人信息？

您的 Elastic Beanstalk 应用程序使用的AWS资源可能存储个人信息。当您终止环境时，Elastic Beanstalk 将终止其创建的资源。还将终止使用配置文件添加的资源。但是，如果您在 Elastic Beanstalk 环境之外创建AWS资源并且将这些资源与应用程序关联，您可能需要手动检查应用程序可能已存储的个人信息是否未被保留。在本开发人员指南中，当我们讨论额外资源的创建时，我们还提到了何时应考虑删除它们。

Elastic Beanstalk 资源

下列相关资源在您使用此服务的过程中会有所帮助。

- Elastic Beanstalk API 参考 所有 SOAP 和查询 API 的全面描述。而且，它还包含一个所有 SOAP 数据类型的列表。
- elastic-beanstalk-samples on GitHub — 包含 Elastic Beanstalk 示例配置文件（.ebextensions）的 GitHub 存储库。存储库`README.md`的文件包含指向包含示例应用程序的其他 GitHub 存储库的链接。
- Elastic Beanstalk 技术常见问题 - 开发人员对此产品提出的一些最热门的问题。
- AWS Elastic Beanstalk 发行说明 — 有关 Elastic Beanstalk 服务、平台、控制台和 EB CLI 版本中的新功能、更新和修复的详细信息。

- 课程和研讨会 — 指向基于角色的课程和专业课程的链接，以及自定进度的实验室，可帮助您提高 AWS 技能并获得实践经验。
- AWS 开发者中心 — 浏览教程、下载工具并了解 AWS 开发者活动。
- AWS 开发者工具-指向开发者工具、SDK、IDE 工具包和命令行工具的链接，用于开发和管理 AWS 应用程序。
- 入门资源中心 — 了解如何设置你的 AWS 账户、加入 AWS 社区和启动你的第一个应用程序。
- 动手教 step-by-step 程 — 按照教程启动您的第一个应用程序 AWS。
- AWS 白皮书 — 指向技术 AWS 白皮书完整列表的链接，这些白皮书涵盖架构、安全和经济学等主题，由 AWS 解决方案架构师或其他技术专家撰写。
- AWS Support 中心 — 创建和管理 AWS Support 案例的中心。还包括指向其他有用资源的链接，例如论坛、技术常见问题解答、服务运行状况和 AWS Trusted Advisor。
- AWS Support— 提供有关 AWS Support快速响应支持渠道信息的主要网页 one-on-one，该渠道可帮助您在云中构建和运行应用程序。
- 联系我们 – 用于查询有关 AWS 账单、账户、事件、滥用和其他问题的中央联系点。
- AWS 网站条款 — 有关我们的版权和商标、您的帐户、许可证和网站访问权限以及其他主题的详细信息。

示例应用程序

以下是在开始使用 Elastic Beanstalk中部署的示例应用程序的下载链接。

> **Note**
>
> 有些示例使用的功能可能是在您使用的平台发布之后发布的。如果示例无法运行，请尝试将您的平台更新为当前版本，如the section called “支持的平台”中所述。

- Docker – docker.zip
- 多容器 Docker — 2.zip docker-multicontainer-v
- 预配置的 Docker (Glassfish) — 1.zip docker-glassfish-v
- Go – go.zip
- Corretto – corretto.zip
- Tomcat – tomcat.zip
- Linux 上的.NET 内核 — dotnet-core-linux.zi
- .NET 核心 — dotnet-asp-windows.zip
- Node.js – nodejs.zip
- PHP – php.zip
- Python – python.zip
- Ruby – ruby.zip

平台历史记录

AWS Elastic Beanstalk 平台历史记录已移动。请参阅 AWS Elastic Beanstalk 平台文档中的平台历史记录。

主题

- Elastic Beanstalk 自定义平台

Elastic Beanstalk 自定义平台

> **Note**
>
> 2022 年 7 月 18 日，Elastic Beanstalk 将基于 Amazon Linux AMI（AL1）的所有平台分支的状态设置为已停用。这包括自定义平台。Elastic Beanstalk 不支持自定义平台。有关停用 Amazon Linux AMI 的 Elastic Beanstalk 的更多信息，请参阅 平台停用常见问题。

本文档中保留了此主题，供在 ElasticBeanstalk 自定义平台功能停用之前使用该功能的客户参考。过去，Elastic Beanstalk 自定义平台支持从 Amazon Linux AMI、RHEL 7、RHEL 6 或 Ubuntu 16.04 基础 AMI 构建 AMI。Elastic Beanstalk 不再支持这些操作系统。若要详细了解不再支持的自定义平台功能，请展开以下主题。

自定义平台

从很多方面而言，自定义平台都是比自定义映像更高级的自定义方式。您可通过自定义平台从头开始开发整个新平台，自定义 Elastic Beanstalk 在平台实例上运行的操作系统、附加软件和脚本。如果 Elastic Beanstalk 并未针对应用程序使用的语言或其他基础设施软件提供托管平台，您也可以灵活地为应用程序构建一个平台。自定义映像是由您修改 Amazon Machine Image (AMI) 以便与现有 Elastic Beanstalk 平台结合使用，而 Elastic Beanstalk 仍旧提供平台脚本，并控制平台的软件堆栈。此外，对于自定义平台，您使用自动化、脚本化的方式创建自定义并进行维护；而自定义映像是您对正在运行的实例进行手动更改。

要创建自定义平台，您需要基于所支持的操作系统 Ubuntu、RHEL 或 Amazon Linux（有关确切版本号，请参阅 Platform.yaml 文件格式中的 `flavor` 条目）之一构建 AMI 并进一步添加自定义。您可使用 Packer（一种开源工具，用于为多种平台创建系统映像，包括用于 Amazon Elastic Compute Cloud

(Amazon EC2) 的 AMI）创建您自己的 Elastic Beanstalk 平台。Elastic Beanstalk 平台包括被配置为运行一组软件以支持应用程序的 AMI，以及可以包含自定义配置选项和默认配置选项设置的元数据。

Elastic Beanstalk 将 Packer 作为单独的内置平台进行管理，您无需担心 Packer 的配置和版本。

您可以通过为 Elastic Beanstalk 提供 Packer 模板、脚本及供模板调用以构建 AMI 的文件来创建平台。这些组件使用指定模板和元数据的平台定义文件打包成一个称作平台定义存档的 ZIP 存档。

在创建自定义平台时，您可以启动单一实例环境，无需运行 Packer 的弹性 IP。Packer 可以随即启动另一个实例来构建映像。您可以对多个平台及每个平台的多个版本重用此环境。

> **Note**
>
> 自定义平台因 AWS 地区而异。如果您需要在多个区域中使用 Elastic Beanstalk，则必须在每个区域中单独创建平台。
>
> 在某些情况下，系统不会清理 Packer 启动的实例，必须手动将其终止。要了解如何手动清理这些实例，请参阅 Packer 实例清理。

您账户中的用户可以通过在创建环境期间指定平台 ARN 来使用您的自定义平台。这些 ARN 由您用于创建自定义平台的 eb platform create 命令返回。

您每次构建自定义平台时，Elastic Beanstalk 都会创建新的平台版本。用户可以通过名称指定平台以只获取此平台的最新版本，也可以包含版本号来获取特定的版本。

例如，要部署 ARN 为 **MyCustomPlatformARN** 的自定义平台的最新版本（可以是 3.0 版），您的 EB CLI 命令行将如下所示：

```
eb create -p MyCustomPlatformARN
```

要部署 2.1 版，您的 EB CLI 命令行将如下所示：

```
eb create -p MyCustomPlatformARN --version 2.1
```

您可以在创建自定义平台版本和编辑现有自定义平台版本的标签时向其应用标签。有关更多信息，请参阅 标记自定义平台版本。

创建自定义平台

要创建自定义平台，应用程序的根目录必须包含一个平台定义 `platform.yaml`，定义用于创建自定义平台的生成器的类型。此文件的格式在Platform.yaml 文件格式中描述。您可以从头开始创建自定义平台，或使用某个示例自定义平台作为起始点。

使用示例自定义平台

创建您自己的自定义平台的一个替代方法是使用一个平台定义存档示例来引导您的自定义平台。您必须先在示例中配置然后才能使用的唯一项目是源 AMI 和区域。

> **Note**
>
> 请勿在生产中使用未经修改的示例自定义平台。示例的目的是展示可用于自定义平台的一些功能，但是它们没有被强化以用于生产环境。

NodePlatform_Ubuntu.zip

此自定义平台基于 Ubuntu 16.04 并支持 Node.js 4.4.4。我们在本节的示例中使用此自定义平台。

NodePlatform_RHEL.zip

此自定义平台基于 RHEL 7.2 并支持 Node.js 4.4.4。

NodePlatform_ AmazonLinux .zip

此自定义平台基于 Amazon Linux 2016.09.1 并支持 Node.js 4.4.4。

TomcatPlatform_Ubuntu.zip

此自定义平台基于 Ubuntu 16.04 并支持 Tomcat 7/Java 8。

CustomPlatform_ NodeSampleApp .zip

一个使用 express 和 ejs 显示静态网页的 Node.js 示例。

CustomPlatform_ TomcatSampleApp .zip

一个在部署时显示静态网页的 Tomcat 示例。

下载示例平台定义存档：`NodePlatform_Ubuntu.zip`。此文件包含平台定义文件、Packer 模板、Packer 在映像创建期间运行的脚本以及 Packer 在平台创建期间复制到生成器实例上的脚本和配置文件。

Example NodePlatform_Ubuntu.zip

```
|-- builder                 Contains files used by Packer to create the custom platform
|-- custom_platform.json    Packer template
|-- platform.yaml           Platform definition file
|-- ReadMe.txt              Briefly describes the sample
```

平台定义文件 `platform.yaml` 告知 Elastic Beanstalk Packer 模板 `custom_platform.json` 的名称。

```
version: "1.0"

provisioner:
  type: packer
  template: custom_platform.json
  flavor: ubuntu1604
```

Packer 模板告诉 Packer 如何针对平台构建 AMI，而且要使用 Ubuntu AMI 作为 HVM 实例类型的平台映像的基础。`provisioners` 部分告知 Packer 将存档中 `builder` 文件夹内的所有文件复制到实例，并在实例上运行 `builder.sh` 脚本。脚本完成后，Packer 从修改的实例创建映像。

Elastic Beanstalk 创建三个可用于在 Packer 中标记 AMI 的环境变量：

AWS_EB_PLATFORM_ARN

自定义平台的 ARN。

AWS_EB_PLATFORM_NAME

自定义平台的名称。

AWS_EB_PLATFORM_VERSION

自定义平台的版本。

示例 `custom_platform.json` 文件使用这些变量来定义它在脚本中使用的以下值：

- `platform_name`，由 `platform.yaml` 设置
- `platform_version`，由 `platform.yaml` 设置
- `platform_arn`，由主生成脚本 `builder.sh` 设置，该脚本显示在示例 `custom_platform.json` 文件的结尾。

custom_platform.json 文件包含两个您必须为其提供值的属性：source_ami 和 region。有关选择正确的 AMI 和区域值的详细信息，请参阅在eb-custom-platforms-samples GitHub 存储库中更新 Packer 模板。

Example custom_platform.json

```
{
  "variables": {
    "platform_name": "{{env `AWS_EB_PLATFORM_NAME`}}",
    "platform_version": "{{env `AWS_EB_PLATFORM_VERSION`}}",
    "platform_arn": "{{env `AWS_EB_PLATFORM_ARN`}}"
  },
  "builders": [
    {
      ...
      "region": "",
      "source_ami": "",
      ...
    }
  ],
  "provisioners": [
    {...},
    {
      "type": "shell",
      "execute_command": "chmod +x {{ .Path }}; {{ .Vars }} sudo {{ .Path }}",
      "scripts": [
        "builder/builder.sh"
      ]
    }
  ]
}
```

根据要对实例进行的修改，您在平台定义存档中包含的脚本和其他文件将大不相同。示例平台包含以下脚本：

- 00-sync-apt.sh – 已注释掉：apt -y update。我们已将此命令注释掉，因为它会提示用户输入信息，这样会中断自动软件包更新。Ubuntu 可能会出现此问题。但是，仍建议运行 apt -y update，这是最佳实践。出于此原因，我们在示例脚本中留下了这条命令作为参考。
- 01-install-nginx.sh – 安装 nginx。
- 02-setup-platform.sh – 安装 wget、tree 和 git。将挂钩和日志记录配置复制到实例，并创建以下目录：

- `/etc/SampleNodePlatform` – 在部署期间上传容器配置文件的位置。
- `/opt/elasticbeanstalk/deploy/appsource/` – 其中 `00-unzip.sh` 脚本在部署期间上传应用程序代码（有关此脚本的信息，请参阅 平台脚本工具 部分）。
- `/var/app/staging/` – 部署期间应用程序源代码在此进行处理。
- `/var/app/current/` – 应用程序源代码处理后在此运行。
- `/var/log/nginx/healthd/` – 增强型运行状况代理将日志写入到此位置。
- `/var/nodejs` – 在部署期间上传 Node.js 文件的位置。

通过 EB CLI 使用示例平台定义存档创建您的第一个自定义平台。

创建自定义平台

1. 安装 EB CLI。
2. 创建一个目录，您将在其中提取示例自定义平台。

```
~$ mkdir ~/custom-platform
```

3. 将 `NodePlatform_Ubuntu.zip` 提取到目录，然后更改为提取的目录。

```
~$ cd ~/custom-platform
~/custom-platform$ unzip ~/NodePlatform_Ubuntu.zip
~/custom-platform$ cd NodePlatform_Ubuntu
```

4. 编辑 `custom_platform.json` 文件，并为 `source_ami` 和 `region` 属性提供值。有关详细信息，请参阅更新 Packer 模板。
5. 运行 eb platform init 并按照提示初始化平台存储库。

   您可以将 eb platform 缩短为 ebp。

> **Note**
>
> Windows PowerShell 使用ebp作为命令别名。如果你在 Windows 中运行 EB CLI PowerShell，请使用以下命令的长格式：eb platform。

```
~/custom-platform$ eb platform init
```

该命令还会在当前目录中创建目录 `.elasticbeanstalk` 并将配置文件 `config.yml` 添加到该目录中。请勿更改或删除此文件，因为 Elastic Beanstalk 在创建自定义平台时依赖此文件。

默认情况下，eb platform init 使用当前文件夹的名称作为自定义平台的名称，在此示例中这将是 `custom-platform`。

6. 运行 eb platform create 以启动 Packer 环境并获取自定义平台的 ARN。稍后您需要此值从自定义平台创建环境。

```
~/custom-platform$ eb platform create
...
```

默认情况下，Elastic Beanstalk 为自定义平台创建实例配置文件 `aws-elasticbeanstalk-custom-platform-ec2-role`。但是，如果您想使用现有实例配置文件，请向 eb platform create 命令中添加选项 `-ip` *`INSTANCE_PROFILE`*。

> **Note**
>
> 如果您使用 Elastic Beanstalk 默认实例配置文件 `aws-elasticbeanstalk-ec2-role`，Packer 将无法创建自定义平台。

EB CLI 会显示 Packer 环境的事件输出，直到构建完成。您可以按 Ctrl+C 来退出事件视图。

7. 您可以使用 eb platform logs 命令检查日志有无错误。

```
~/custom-platform$ eb platform logs
...
```

8. 稍后，您可以使用 eb platform events 来检查过程。

```
~/custom-platform$ eb platform events
...
```

9. 使用 eb platform status 检查平台状态。

```
~/custom-platform$ eb platform status
...
```

操作完成后，您就拥有了一个可以用于启动 Elastic Beanstalk 环境的平台。

在从控制台创建环境时，可以使用自定义平台。请参阅 创建新环境向导。

在您的自定义平台上启动环境

1. 为您的应用程序创建一个目录。

```
~$ mkdir custom-platform-app
~$ cd ~/custom-platform-app
```

2. 初始化应用程序存储库。

```
~/custom-platform-app$ eb init
...
```

3. 下载示例应用程序 NodeSampleApp.zip。
4. 提取示例应用程序。

```
~/custom-platform-app$ unzip ~/NodeSampleApp.zip
```

5. 运行 eb create -p ***CUSTOM-PLATFORM-ARN***，其中 *CUSTOM-PLATFORM-ARN* 是 eb platform create 命令返回的 ARN，用于启动运行您自定义平台的环境。

```
~/custom-platform-app$ eb create -p CUSTOM-PLATFORM-ARN
...
```

平台定义存档内容

平台定义存档是相当于应用程序源包的平台。平台定义存档是一种 ZIP 文件，其中包含平台定义文件、Packer 模板以及 Packer 模板用来创建平台的脚本和文件。

> **Note**
>
> 当您使用 EB CLI 创建自定义平台时，EB CLI 将从平台存储库中的文件和文件夹创建平台定义存档，因此您不需要手动创建存档。

平台定义文件是 YAML 格式的文件，名称必须为 `platform.yaml` 并且位于平台定义存档的根目录中。有关平台定义文件中支持的必需和可选键的列表，请参阅 创建自定义平台。

您不必以特定方式命名 Packer 模板，但文件的名称必须与平台定义文件中指定的配置器模板匹配。有关创建 Packer 模板的说明，请参阅官方 Packer 文档。

您的平台定义存档中的其他文件是模板在创建 AMI 之前自定义实例所要使用的脚本和文件。

自定义平台挂钩

Elastic Beanstalk 在自定义平台上为挂钩使用了标准化的目录结构。挂钩是在生命周期事件期间运行以响应管理操作的脚本，如启动环境中的实例时、当用户启动部署或使用重新启动应用程序服务器功能时。

将您希望挂钩触发的脚本置于 `/opt/elasticbeanstalk/hooks/` 文件夹的一个子文件夹中。

> **Warning**
>
> 不支持在托管平台上使用自定义平台挂钩。自定义平台挂钩是为自定义平台设计的。在 Elastic Beanstalk 托管平台上，它们的行为可能不同或存在一些问题，而且其行为可能因平台而异。在 Amazon Linux AMI 平台（Amazon Linux 2 以前的版本）上，在某些情况下，它们可能仍然有用；请谨慎使用它们。
>
> 自定义平台挂钩是 Amazon Linux AMI 平台上存在的旧功能。在 Amazon Linux 2 平台上，`/opt/elasticbeanstalk/hooks/` 文件夹中的自定义平台挂钩将完全中断。Elastic Beanstalk 不会读取或执行它们。Amazon Linux 2 平台支持一种新型平台挂钩，专门设计用于扩展 Elastic Beanstalk 托管平台。您可以将自定义脚本和程序直接添加到应用程序源包的挂钩目录中。Elastic Beanstalk 在不同实例预置阶段运行它们。有关更多信息，请在 the section called “扩展 Linux 平台” 中展开平台挂钩部分。

挂钩组织到以下文件夹中：

- `appdeploy` - 在应用程序部署期间运行的脚本。Elastic Beanstalk 在启动新实例时以及在客户端启动新版本部署时执行应用程序部署。
- `configdeploy` – 当客户端执行影响实例上的软件配置的配置更新（如通过设置环境属性或启用到 Amazon S3 的日志轮换）时运行的脚本。
- `restartappserver` - 当客户端执行重新启动应用程序服务器操作时运行的脚本。
- `preinit` - 在实例引导启动期间运行的脚本。
- `postinit` - 在实例引导启动后运行的脚本。

appdeploy、configdeploy 和 restartappserver 文件夹包含 pre、enact 和 post 子文件夹。在操作的每个阶段，pre 文件夹、enact 文件夹和 post 文件夹中的所有脚本依次按字母顺序运行。

在启动实例时，Elastic Beanstalk 会按顺序运行 preinit、appdeploy 和 postinit。在到正在运行的实例的后续部署中，Elastic Beanstalk 会运行 appdeploy 挂钩。configdeploy 挂钩在用户更新实例软件配置设置时运行。restartappserver 挂钩只在用户执行应用程序服务器重新启动时运行。

当脚本遇到错误时，它们能以非零状态退出并写入 stderr，以使操作失败。写入 stderr 的消息将显示在操作失败时输出的事件中。Elastic Beanstalk 还会将此信息捕获到日志文件 /var/log/eb-activity.log 中。如果您不想使操作失败，请返回 0（零）。写入 stderr 或 stdout 的消息会显示在部署日志中，但除非操作失败，否则不会显示在事件流中。

Packer 实例清理

在某些情况下，例如在 Packer 生成器进程完成之前将其停止，Packer 启动的实例将不会被清理。这些实例不是 Elastic Beanstalk 环境的一部分，只能使用 Amazon EC2 服务查看和终止。

手动清理这些实例

1. 打开 Amazon EC2 控制台。
2. 确保您所在的 AWS 区域与使用 Packer 创建实例的区域相同。
3. 在 Resources（资源）下，选择 *N* 个正在运行的实例，其中 *N* 指示正在运行的实例数量。
4. 在查询文本框中单击。
5. 选择 Name (名称) 标签。
6. 输入 packer。

   查询应类似于：tag:Name: packer
7. 选择符合查询条件的任何实例。
8. 如果实例状态是正在运行，请依次选择操作、实例状态、停止，然后再依次选择操作、实例状态、终止。

Platform.yaml 文件格式

platform.yaml 文件采用以下格式。

```
version: "version-number"
```

```
provisioner:
   type: provisioner-type
   template: provisioner-template
   flavor: provisioner-flavor

metadata:
   maintainer: metadata-maintainer
   description: metadata-description
   operating_system_name: metadata-operating_system_name
   operating_system_version: metadata-operating_system_version
   programming_language_name: metadata-programming_language_name
   programming_language_version: metadata-programming_language_version
   framework_name: metadata-framework_name
   framework_version: metadata-framework_version

option_definitions:
   - namespace: option-def-namespace
     option_name: option-def-option_name
     description: option-def-description
     default_value: option-def-default_value

option_settings:
   - namespace: "option-setting-namespace"
     option_name: "option-setting-option_name"
     value: "option-setting-value"
```

使用以下值替换占位符：

version-number

必需。YAML 定义的版本。必须是 **1.0**。

provisioner-type

必需。用于创建自定义平台的生成器的类型。必须是 **packer**。

provisioner-template

必需。包含 *provisioner-type* 设置的 JSON 文件。

provisioner-flavor

可选。用于 AMI 的基本操作系统。下列情况之一：

amazon (默认)

Amazon Linux。如果未指定，则是创建该平台时的 Amazon Linux 的最新版本。

Amazon Linux 2 不是支持的操作系统版本。

ubuntu1604

Ubuntu 16.04 LTS

rhel7

RHEL 7

rhel6

RHEL 6

metadata-maintainer

可选。平台所有者的联系信息 (100 个字符)。

metadata-description

可选。平台的描述 (2000 个字符)。

metadata-operating_system_name

可选。平台的操作系统的名称 (50 个字符)。在筛选 ListPlatformVersionsAPI 的输出时，此值可用。

metadata-operating_system_version

可选。平台的操作系统的版本 (20 个字符)。

metadata-programming_language_name

可选。平台支持的编程语言 (50 个字符)

metadata-programming_language_version

可选。平台的语言的版本 (20 个字符)。

metadata-framework_name

可选。平台使用的 Web 框架的名称 (50 个字符)。

metadata-framework_version

可选。平台的 Web 框架的版本 (20 个字符)。

option-def-namespace

可选。aws:elasticbeanstalk:container:custom 下的命名空间（100 个字符）。

option-def-option_##

可选。选项的名称 (100 个字符)。您可以定义最多 50 个由平台向用户提供的自定义配置选项。

option-def-description

可选。选项的描述 (1024 个字符)。

option-def-default_value

可选。在用户未指定值时使用的默认值。

以下示例将创建选项 **NPM_START**。

```
options_definitions:
 -  namespace: "aws:elasticbeanstalk:container:custom:application"
    option_name: "NPM_START"
    description: "Default application startup command"
    default_value: "node application.js"
```

option-setting-namespace

可选。选项的命名空间。

option-setting-option_##

可选。选项的名称。您可以指定最多 50 个由 Elastic Beanstalk 提供的选项。

option-setting-value

可选。在用户未指定值时使用的值。

以下示例将创建选项 **TEST**。

```
option_settings:
 - namespace: "aws:elasticbeanstalk:application:environment"
   option_name: "TEST"
   value: "This is a test"
```

标记自定义平台版本

您可以将标签应用于您的 AWS Elastic Beanstalk 自定义平台版本。标签是与资源关联的键值对。AWS 有关 Elastic Beanstalk 资源标记、使用案例、标签键和值约束以及支持的资源类型的信息，请参阅标记 Elastic Beanstalk 应用程序资源。

您可以在创建自定义平台版本时指定标签。在现有自定义平台版本中，您可以添加或删除标签，以及更新现有标签的值。您最多可以为每个自定义平台版本添加 50 个标签。

在自定义平台版本创建期间添加标签

如果使用 EB CLI 创建自定义平台版本，则可以使用 eb platform create 的 `--tags` 选项添加标签。

```
~/workspace/my-app$ eb platform create --tags mytag1=value1,mytag2=value2
```

对于 AWS CLI 或其他基于 API 的客户端，使用命令中的`--tags`参数添加标签。create-platform-version

```
$ aws elasticbeanstalk create-platform-version \
      --tags Key=mytag1,Value=value1 Key=mytag2,Value=value2 \
      --platform-name my-platform --platform-version 1.0.0 --platform-definition-bundle
 S3Bucket=DOC-EXAMPLE-BUCKET,S3Key=sample.zip
```

管理现有自定义平台版本的标签

您可以在现有 Elastic Beanstalk 自定义平台版本中添加、更新和删除标签。

如果使用 EB CLI 更新自定义平台版本，则可使用 eb tags 来添加、更新、删除或列出标签。

例如，以下命令会列出自定义平台版本中的标签。

```
~/workspace/my-app$ eb tags --list --resource "arn:aws:elasticbeanstalk:us-east-2:my-
account-id:platform/my-platform/1.0.0"
```

以下命令会更新标签 `mytag1` 并删除标签 `mytag2`。

```
~/workspace/my-app$ eb tags --update mytag1=newvalue --delete mytag2 \
      --resource "arn:aws:elasticbeanstalk:us-east-2:my-account-id:platform/my-
platform/1.0.0"
```

有关选项和更多示例的完整列表，请参阅 `eb tags`。

对于 AWS CLI 或其他基于 API 的客户端，使用list-tags-for-resource命令列出自定义平台版本的标签。

```
$ aws elasticbeanstalk list-tags-for-resource --resource-arn
 "arn:aws:elasticbeanstalk:us-east-2:my-account-id:platform/my-platform/1.0.0"
```

使用 update-tags-for-resource 命令可在自定义平台版本中添加、更新或删除标签。

```
$ aws elasticbeanstalk update-tags-for-resource \
      --tags-to-add Key=mytag1,Value=newvalue --tags-to-remove mytag2 \
      --resource-arn "arn:aws:elasticbeanstalk:us-east-2:my-account-id:platform/my-
platform/1.0.0"
```

在 `--tags-to-add` 的 update-tags-for-resource 参数中指定要添加的标签和要更新的标签。添加了一个不存在的标签，更新了现有标签的值。

> **Note**
>
> 要在 Elastic Beanstalk 自定义平台版本中使用某些 EB CLI 和 AWS CLI 命令，您需要自定义平台版本的 ARN。您可以使用下面的命令检索该 ARN。
>
> ```
> $ aws elasticbeanstalk list-platform-versions
> ```
>
> 使用 `--filters` 选项从输出筛选出自定义平台的名称。

文档历史记录

下表描述了自 2024 年 4 月以来对《AWS Elastic Beanstalk 开发者指南》所做的重要更改。

变更	说明	日期
QuickStart 适用于 Windows 上的.NET 核心版	Windows 上.NET Core 的新增 QuickStart 功能。	2024年6月28日
QuickStart 适用于 Docker	Docker QuickStart 的新增功能。	2024年6月19日
防止跨环境访问 Amazon S3 存储桶	新增防止跨环境访问 Amazon S3 存储桶。	2024 年 6 月 12 日
QuickStart 适用于 Linux 上的.NET	Windows 上.NET Core 的新增 QuickStart 功能。	2024 年 5 月 28 日
QuickStart 对于 PHP	PHP QuickStart 的新增功能。	2024年5月10日
QuickStart 对于 Node.js	Node.js QuickStart 的新增功能。	2024年5月5日
QuickStart for Go	Go QuickStart 新增功能。	2024年5月5日
Elastic Beanstalk 平台发布时间表	添加了一个新主题，其中包含的时间表即将发布的平台分支版本。移停用平台分支计划到了这个已停用平台分支历史记录话题上。	2024 年 5 月 1 日
AWSElasticBeanstalkRoleCore AWS 托管策略	更新了 AWS 托管策略中的权限。	2024 年 4 月 30 日
AWSElasticBeanstalkManagedUpdatesServiceRolePolicy AWS 托管策略	更新了 AWS 托管策略中的权限。	2024 年 4 月 30 日

AWSElasticBeanstalkManagedUpdatesInternalServiceRolePolicy AWS 托管策略	更新了 AWS 托管策略中的权限。	2024 年 4 月 30 日
AWSElasticBeanstalkMaintenance AWS 托管策略	更新了 AWS 托管策略中的权限。	2024 年 4 月 30 日
AWSElasticBeanstalkInternalMaintenanceRolePolicy AWS 托管策略	更新了 AWS 托管策略中的权限。	2024 年 4 月 30 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。